A0597203 AI Business Applications

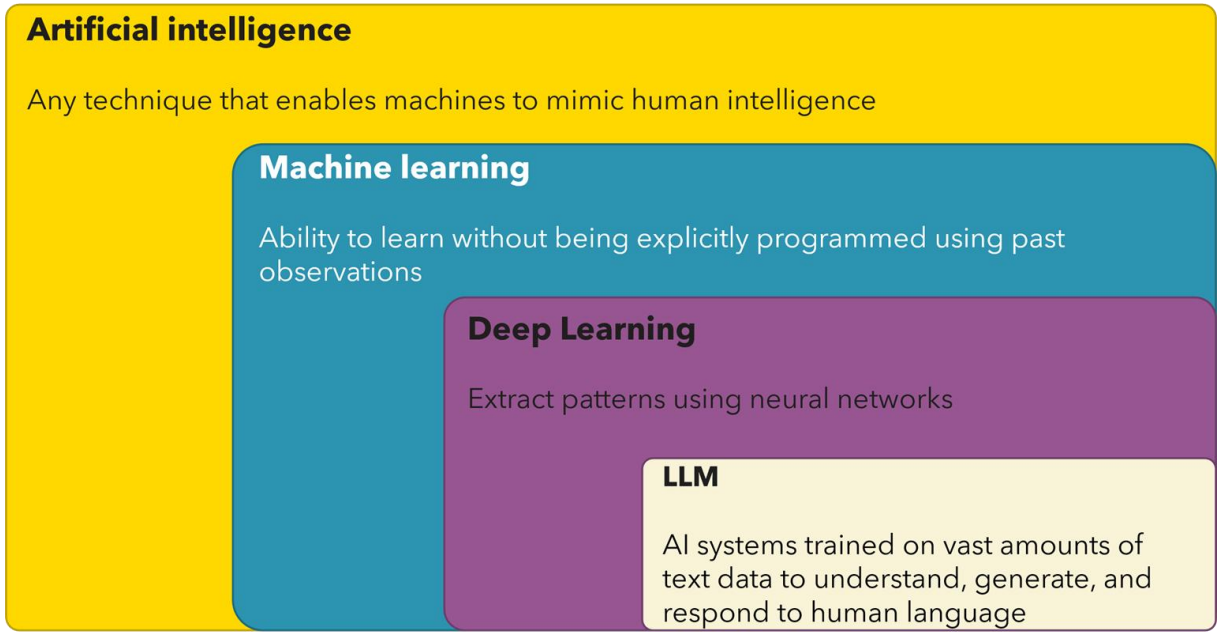
Introduction to Generative AI (GenAI)

https://www.knime.com/events/genai-data-workflows-getting-started-course

Introduction to Generative AI

The AI Landscape: A Quick Recap

In our previous sessions, we explored two major categories of artificial intelligence:



Supervised Learning (Predictive AI):

- This is like giving the AI a spreadsheet with questions and answers.
- The AI learns to predict the answer to a new question based on the examples.
- Think of an AI that classifies customer emails as "urgent" or "not urgent" based on past, labeled examples.

Unsupervised Learning (Descriptive AI):

- Here, we give the AI a dataset without any labels or answers.
- The AI's job is to find hidden patterns and relationships on its own.
- A good example is an AI that groups customers into different segments based on their purchasing behavior, helping a business understand its market better.

These two types of AI are primarily focused on analyzing and understanding existing data. They classify, predict, or find patterns.

What is Generative AI?

Generative AI is a fundamentally different class of AI.

Instead of just analyzing data, it's designed to **create new, original content** that has never existed before. This content can take many forms:

- **Text:** Articles, emails, code, poems, scripts, and marketing copy.
- **Images:** Photos, digital art, logos, and product designs.
- **Audio:** Music, voiceovers, and sound effects.
- **Video:** Short clips, animations, and movie scenes.

Why is it called "Generative"?

- The term "generative" simply means "capable of producing or creating."
- The AI doesn't just retrieve information; it synthesizes it in new and imaginative ways to generate something novel.
- This is a massive leap forward, as it moves AI from a tool for analysis to a partner for **creation and innovation**.

Why Does Generative AI Matter for Business?

Generative AI is a powerful tool for driving business value. Its ability to create content can:

- **Increase Efficiency:** It can automate time-consuming, repetitive tasks like drafting emails, writing product descriptions, or summarizing documents, allowing employees to focus on higher-value work.
- **Unlock Creativity and Innovation:** It can act as a brainstorming partner, generating new ideas for products, marketing campaigns, or even business strategies. This can accelerate the creative process from days to minutes.
- **Personalize at Scale:** Businesses can use it to create highly personalized content for individual customers, such as customized marketing emails or product recommendations, at a massive scale that was previously impossible.
- In essence, generative AI is a new type of intelligent automation that can handle complex, creative tasks.
- For a business leader, understanding how to apply this technology is key to gaining a competitive advantage.

Large Language Models (LLMs)

Large Language Models (LLMs)

- Currently, the most common form of generative AI is a **Large Language Model**, or **LLM**.
- **LLMs** are the **engines** that power popular tools like ChatGPT, Gemini, and Claude.
- Large Language Model is a massive neural network that has been trained on an enormous amount of text and data from the internet—books, articles, websites, and more.
- This training process isn't about memorizing facts; it's about learning the **patterns, grammar, and relationships** within language.
- These models can understand and generate human language, enabling tasks like answering questions, summarizing text, or writing content.

History of Neural Networks

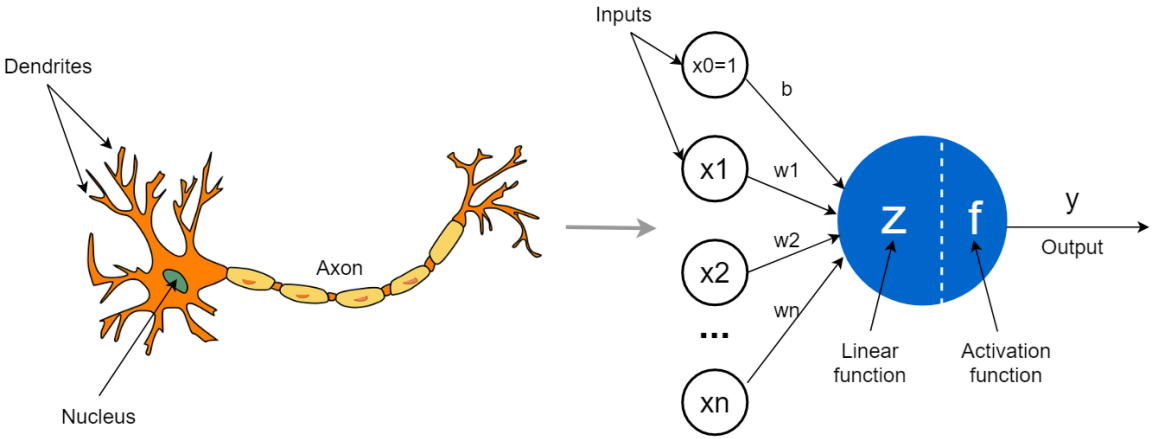
- To understand how LLMs work, it helps to understand some fundamentals of neural networks, as they are the foundation of many AI systems.
- Neural networks are inspired by the human brain and consist of layers of interconnected 'neurons' that learn patterns in data by adjusting connection strengths during training.
- In 1943, researchers Warren McCullock and published their first concept of simplified brain cell.
- This was called McCullock-Pitts (MCP) neuron.
- They described such a nerve cell as a simple logic gate with binary outputs.
- Multiple signals arrive at the dendrites and are then integrated into the cell body, and, if the accumulated signal exceeds a certain threshold, an output signal is generated that will be passed on by the axon.

Warren Sturgis McCulloch

(1898 – 1969)

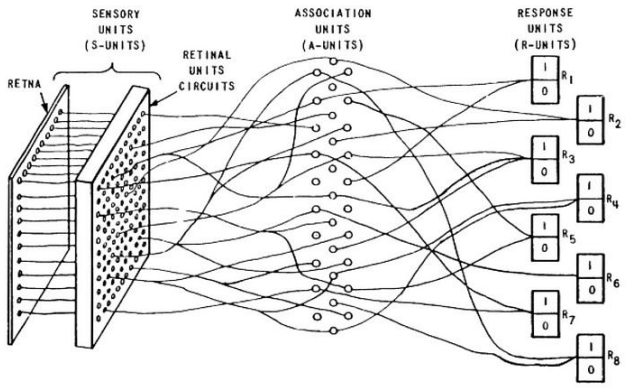
Walter Harry Pitts, Jr.

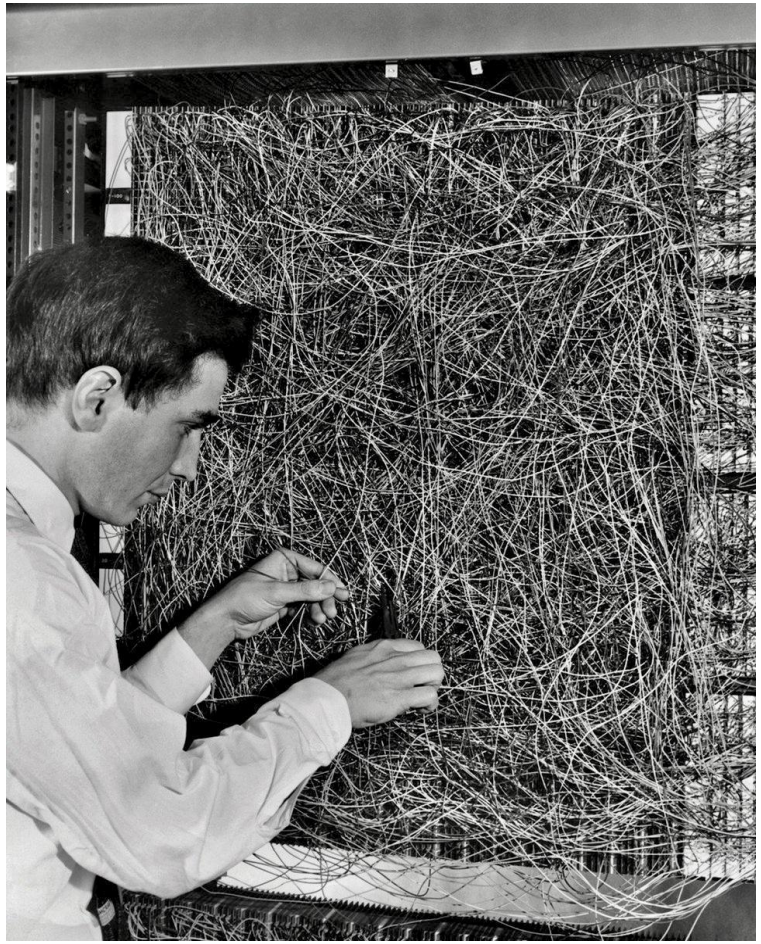
(1923 – 1969)



The Perceptron: Building Block of Neural Networks

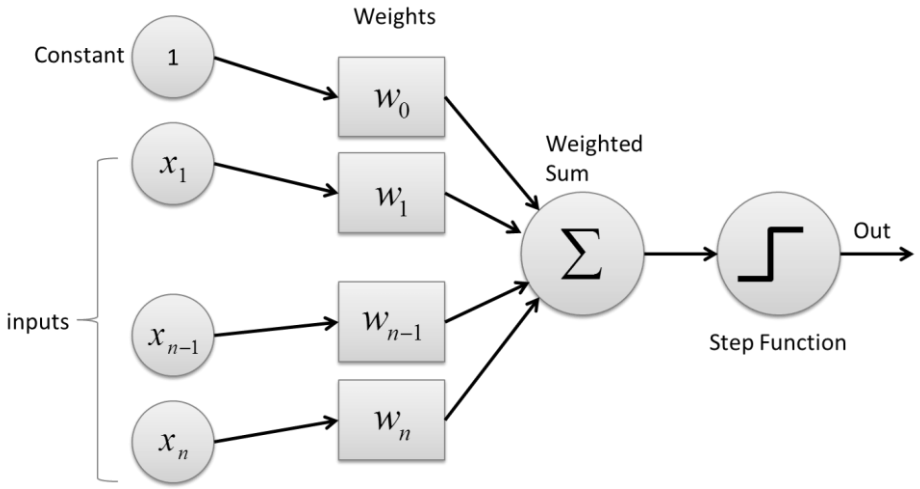
- In 1953, inspired by McCullock work, Frank Rosenblatt invented the **Perceptron**.
- The Perceptron is the simplest form of a neural network.
- It can serve as a binary classifier (separates data into two categories).





The Components of the Perceptron

- Inputs: $x_1, x_2, \ldots, x_n$
- Weights: $w_1, w_2, \ldots, w_n$
- Bias: b
- Activation function: Step function
- Output: 1 if weighted sum > threshold, 0 otherwise

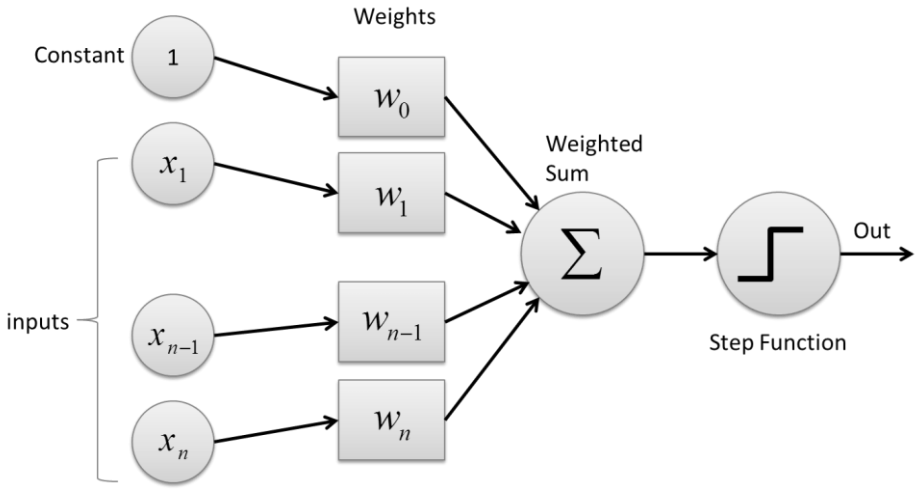


How a Perceptron Works

1. Multiply each input by its corresponding weight
2. Sum all weighted inputs
3. Add the bias term
4. Apply the activation function
5. Output the result

Mathematically:

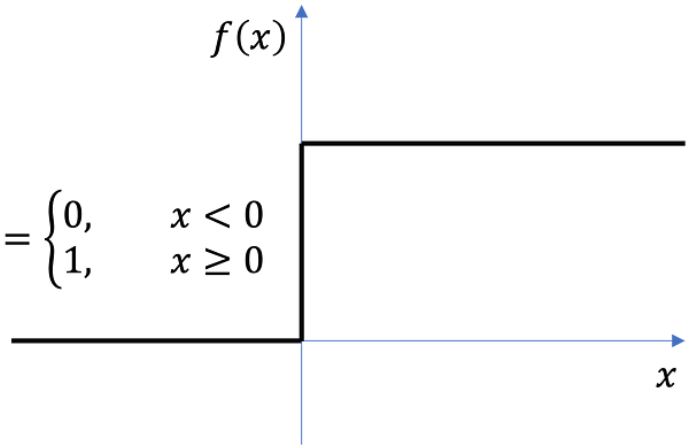
- $z = w_1x_1 + w_2x_2 + \ldots + w_nx_n + b$
- output = activation(z)



Perceptron Activation Function

- **Step Function**:
  - Output: 1 if z ≥ 0, 0 if z < 0
  - Used in the original perceptron
  - Not differentiable at 0

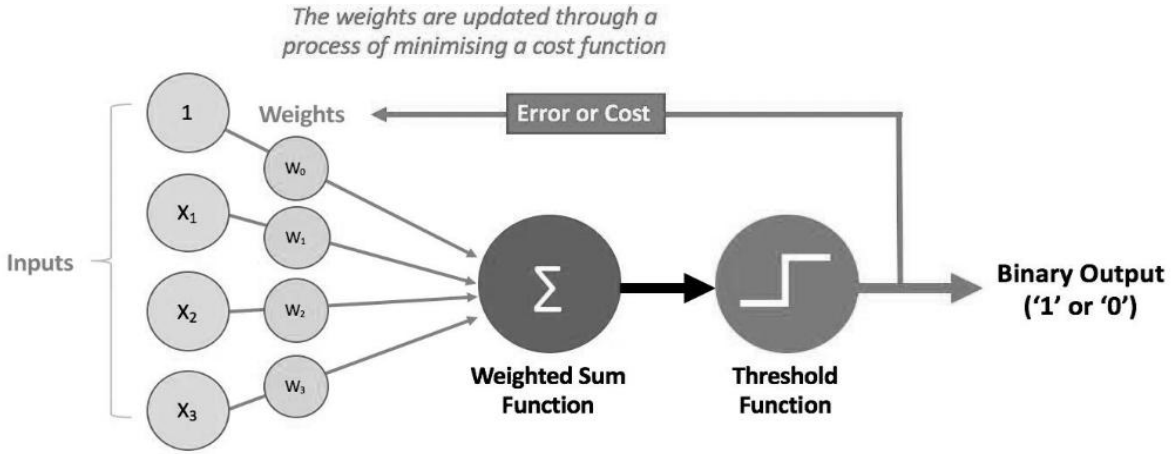
$$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$$



How Perceptron Learn (The Cost Function)

For each training example:

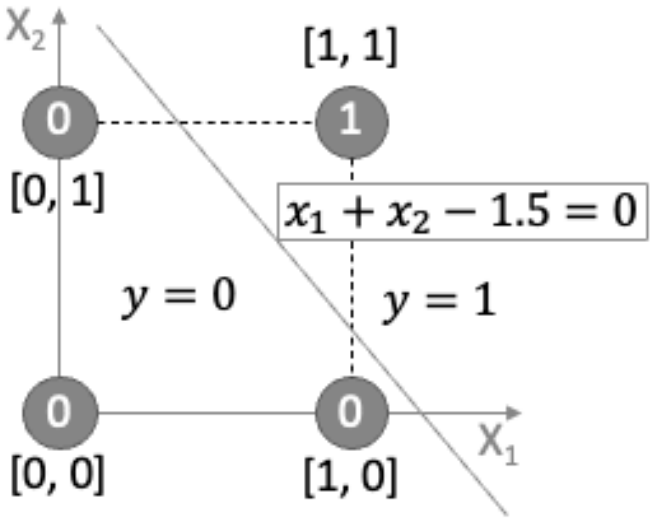
1. Calculate predicted output y_pred
2. Calculate error: error = y_true - y_pred
3. Update weights: w_new = w_old + learning_rate * error * x
4. Update bias: b_new = b_old + learning_rate * error



Decision Boundary

- The perceptron learns a decision boundary: $w_1x_1 + w_2x_2 + b = 0$
- Points above the line are classified as 1
- Points below the line are classified as 0
- For AND gate, only the point (1,1) should be above the line

X_1	X_2	y
0	0	0
0	1	0
1	0	0
1	1	1



Limitations of Simple Perceptron

- Can only learn linearly separable patterns
- Cannot solve XOR problem (need multiple layers)
- No probabilistic output
- Simple update rule isn't suitable for complex problems

AND Truth Table

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

OR Truth Table

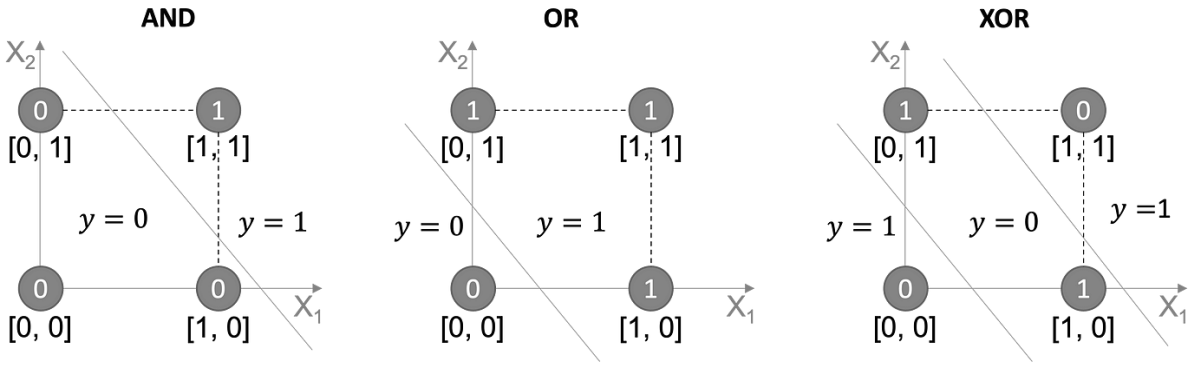
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

XOR Truth Table

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

NOT Truth Table

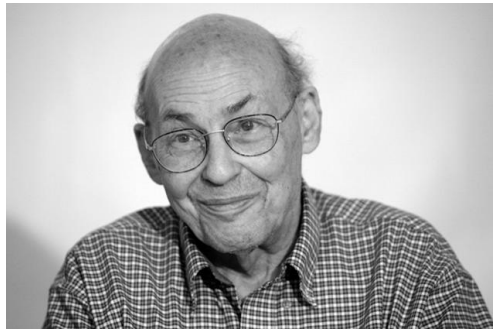
A	B
0	1
1	0



Introduction to Neural Networks

Multi-Layer Neural Networks

The Multi-Layer Percecptron (MLP)



Limitations of the Perceptron

While useful for linearly separable problems, the single perceptron cannot solve complex problems like XOR classification, as demonstrated by Minsky and Papert in their 1969 book "Perceptrons.“

The Multi-Layer Perceptron

The Multi-Layer Perceptron addresses the limitations of the single perceptron by introducing:

- Multiple layers of neurons
- Non-linear activation functions
- More sophisticated learning algorithms

Structure of an MLP

Definition: An MLP is a class of feedforward artificial neural network that consists of at least three layers of nodes: **input**, **hidden**, and **output** layers.

Key Feature: Each neuron in one layer is connected to every neuron in the next layer (fully connected).

1. Input Layer

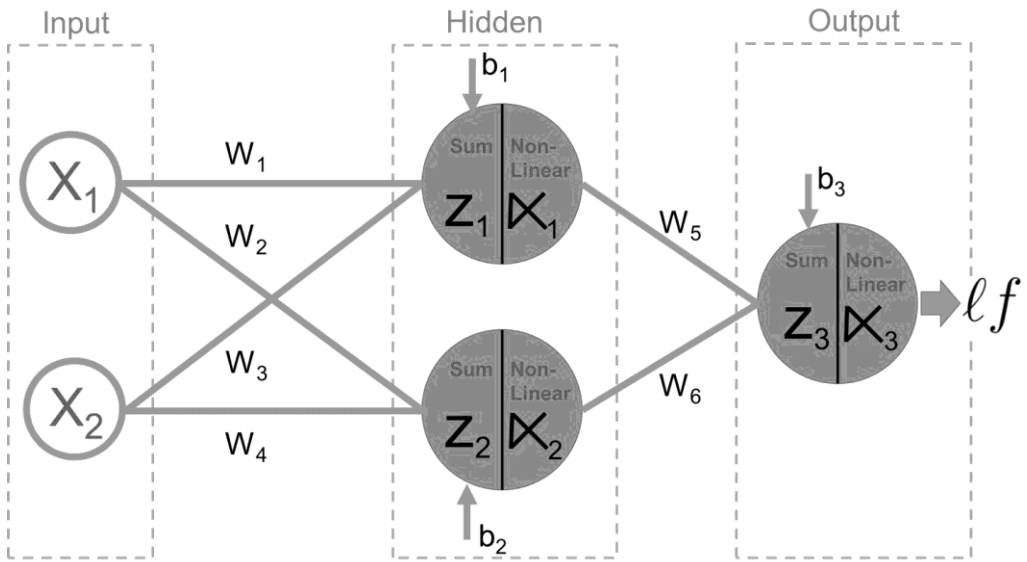
- Receives the raw input features
- One neuron per input feature
- No computation occurs here; inputs are simply passed forward

2. Hidden Layer(s)

- One or more layers between input and output
- Each neuron in a hidden layer:
- Receives inputs from all neurons in the previous layer
- Computes a weighted sum
- Applies a non-linear activation function
- Passes the result to the next layer

3. Output Layer

- Produces the final prediction or classification
- Structure depends on the task:
  - Regression: Often a single neuron with linear activation
  - Binary classification: One neuron with sigmoid activation
  - Multi-class classification: Multiple neurons (one per class) with softmax activation



Structure of an MLP

3. Neurons and Connections

- Each neuron computes a weighted sum of inputs and applies an activation function
- Fully connected between layers (dense connections)

4. Activation Functions

- Introduce non-linearity
- Examples: ReLU, Sigmoid, Tanh, Softmax

5. Loss Function

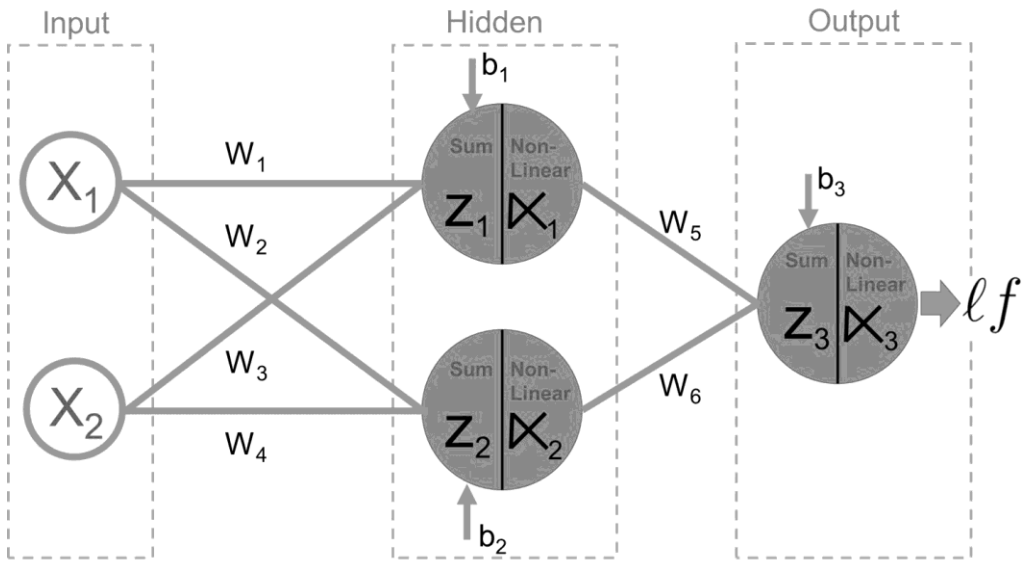
- Measures the error between predicted and true outputs
- Examples: Mean Squared Error, Cross-Entropy

6. Optimizer

- Updates weights to minimize loss
- Examples: SGD, Adam

7. Training Data

- Labeled data used to train the network
- Split into training, validation, and test sets



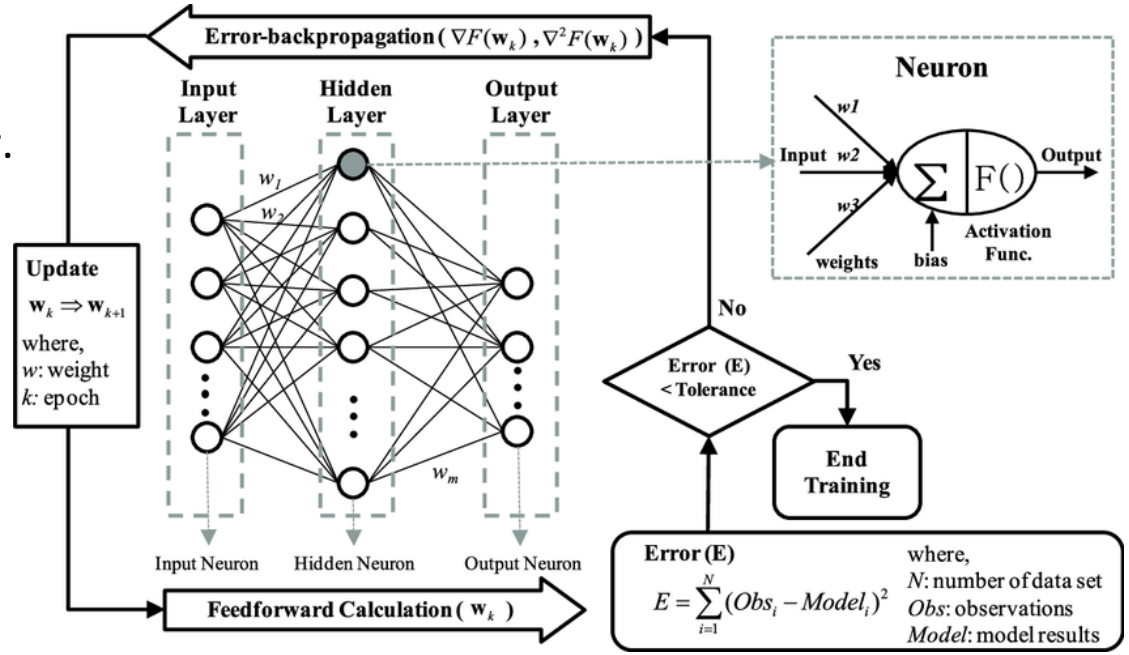
How an MLP Learn – Step by Step

Forward Propagation

- Input features pass through the network layer by layer.
- Each neuron computes a weighted sum of inputs and applies an activation function.
- The final layer produces a prediction.

Loss Computation

- A loss function measures the difference between predicted and actual outputs.
- Common loss functions:
  - Mean Squared Error (regression)
  - Cross-Entropy Loss (classification)



How an MLP Learn – Training Process

Backpropagation

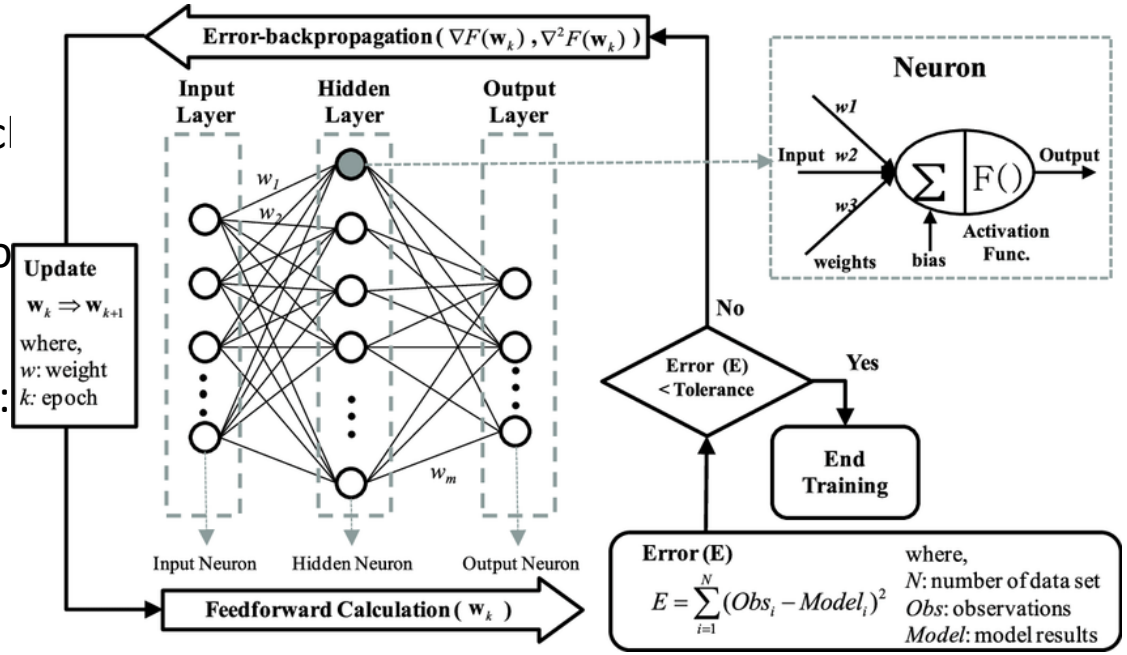
- Gradients of the loss are calculated with respect to eac weight using the chain rule.
- This identifies how each weight contributed to the erro

Weight Update

- An optimizer (e.g., SGD) adjusts weights to reduce loss:
  - w := w - learning_rate × gradient
- This process repeats over multiple iterations (epochs) using training data.

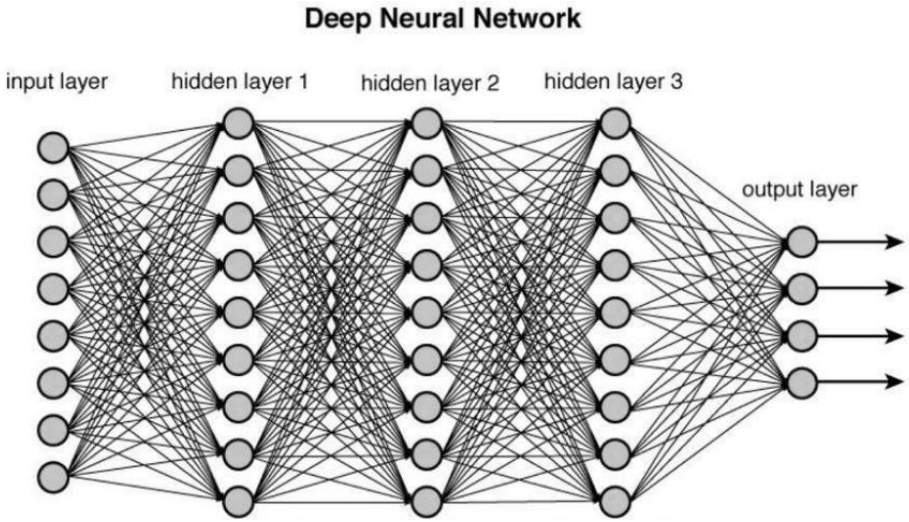
Goal

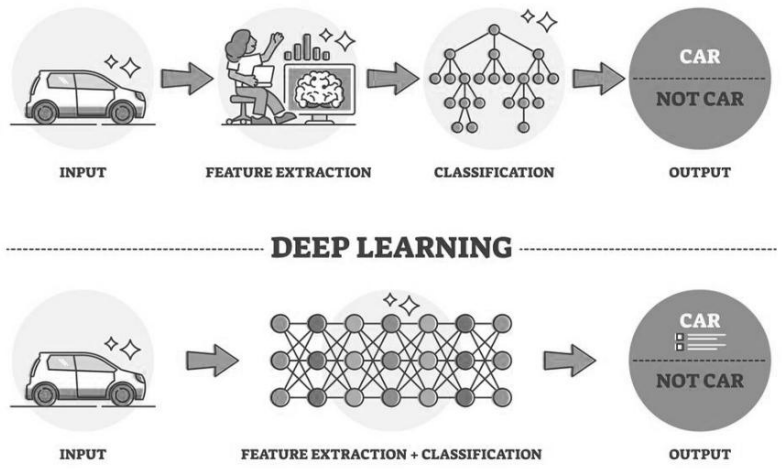
- Gradually minimize the loss and improve prediction accuracy.
- **Loss Function**: MSE for regression, Cross-Entropy for classification.
- **Optimization**: Backpropagation + Gradient Descent (or Adam).



Introduction to Deep Learning

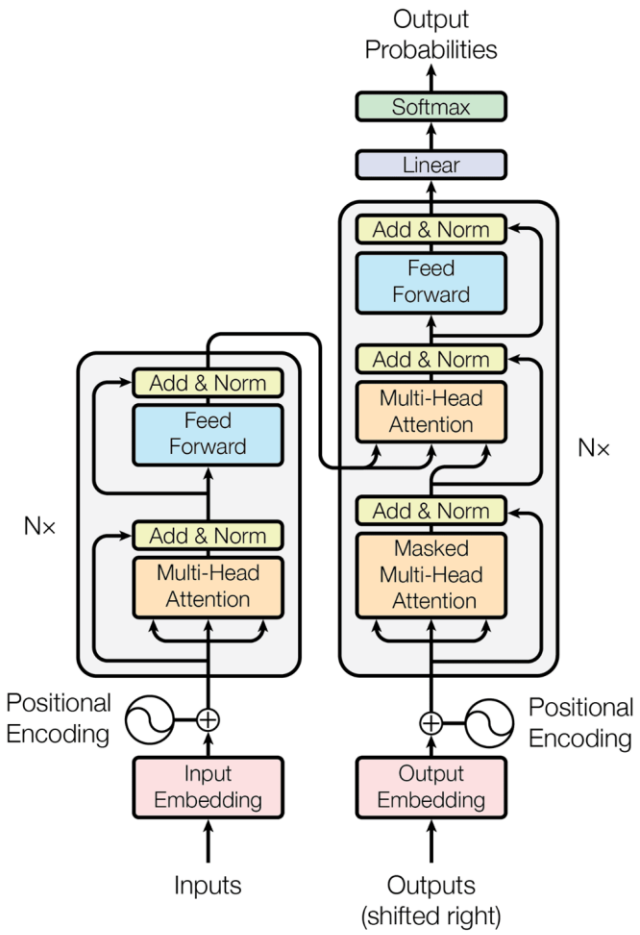
- Neural Networks have revolutionized artificial intelligence by enabling machines to learn from data in ways that mimic human neural processes.
- Deep neural networks (DNNs) are Neural Networks that are composed of multiple processing layers that can learn representations of data with multiple levels of abstraction.
- The power of deep learning comes from its ability to automatically discover intricate patterns in raw data through the learning process, without requiring human engineers to manually specify all the knowledge needed by the computer system.





Large Language Models (LLMs)

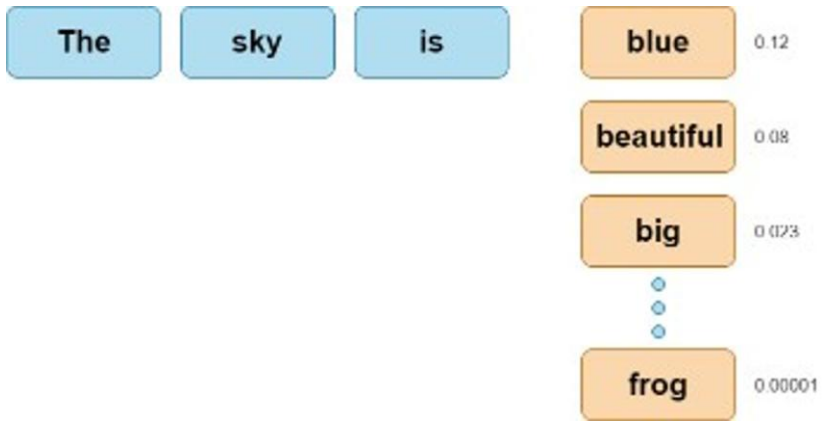
- Large Language Models are built on the Transformer Architecture, which is a neural network architecture that is well-suited for processing sequences.
- These models are trained to learn patterns in human language, enabling them to generate coherent text and support more natural, intuitive interactions with technology.



How do LLMs work?

- LLMs function as complex auto-completion systems.
- They are designed to suggest the most likely next word based on their exposure to similar contexts during training.
- Consider this example:

  when given the phrase "The sky is", an LLM predicts the most likely next word based on learned patterns.
- "Blue" might rank highest, while words like "beautiful" or "big" are also plausible.
- Less likely words, like "frog", may still receive some probability, even if they don't fit the context.



- LLMs operate based on probabilistic knowledge, predicting words and phrases according to patterns in data rather than true understanding.
- They do not possess a semantic understanding of the content they generate.

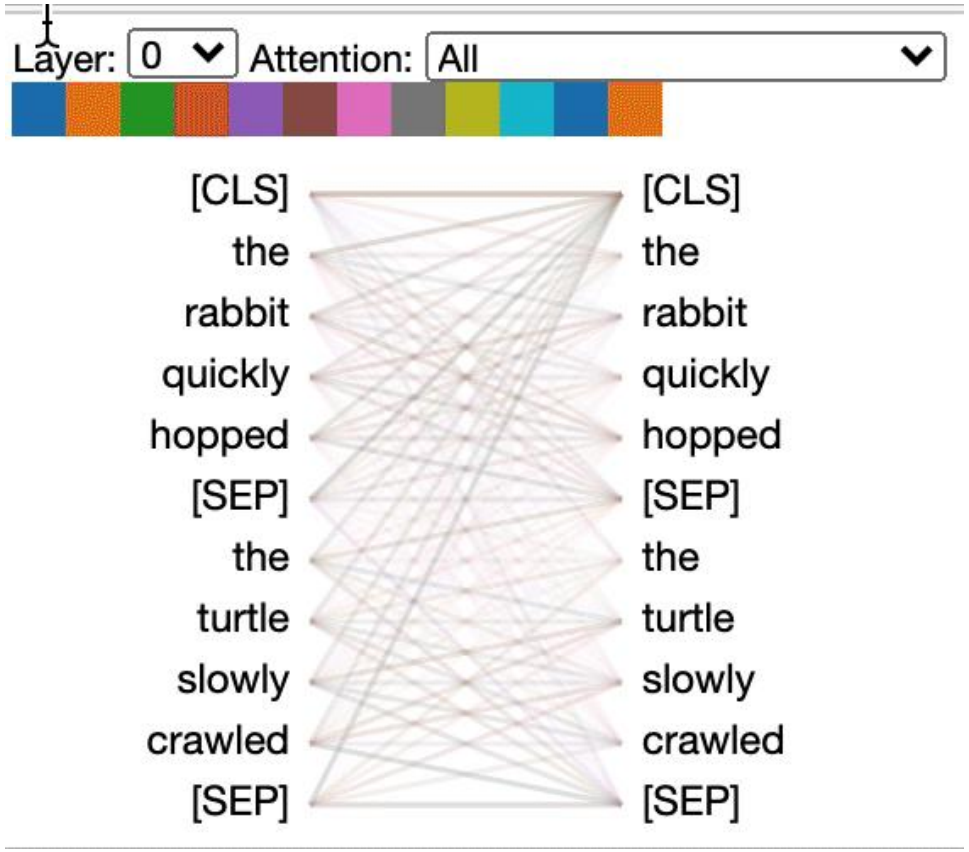
YouTube

Self Attention Mechanism

- Self-Attention allows each word in a sequence to focus on other relevant words.
- It computes a **weighted representation** of each element based on its relationships with others.
- Helps capture **context and dependencies**, even for distant words in a sentence.
- Used in **Transformers (e.g., BERT, GPT)** to understand meaning in context.

Example:

In *"The rabbit quickly hopped"*, the word **"quickly"** attends strongly to **"hopped"**, showing relevance between an adverb and verb.

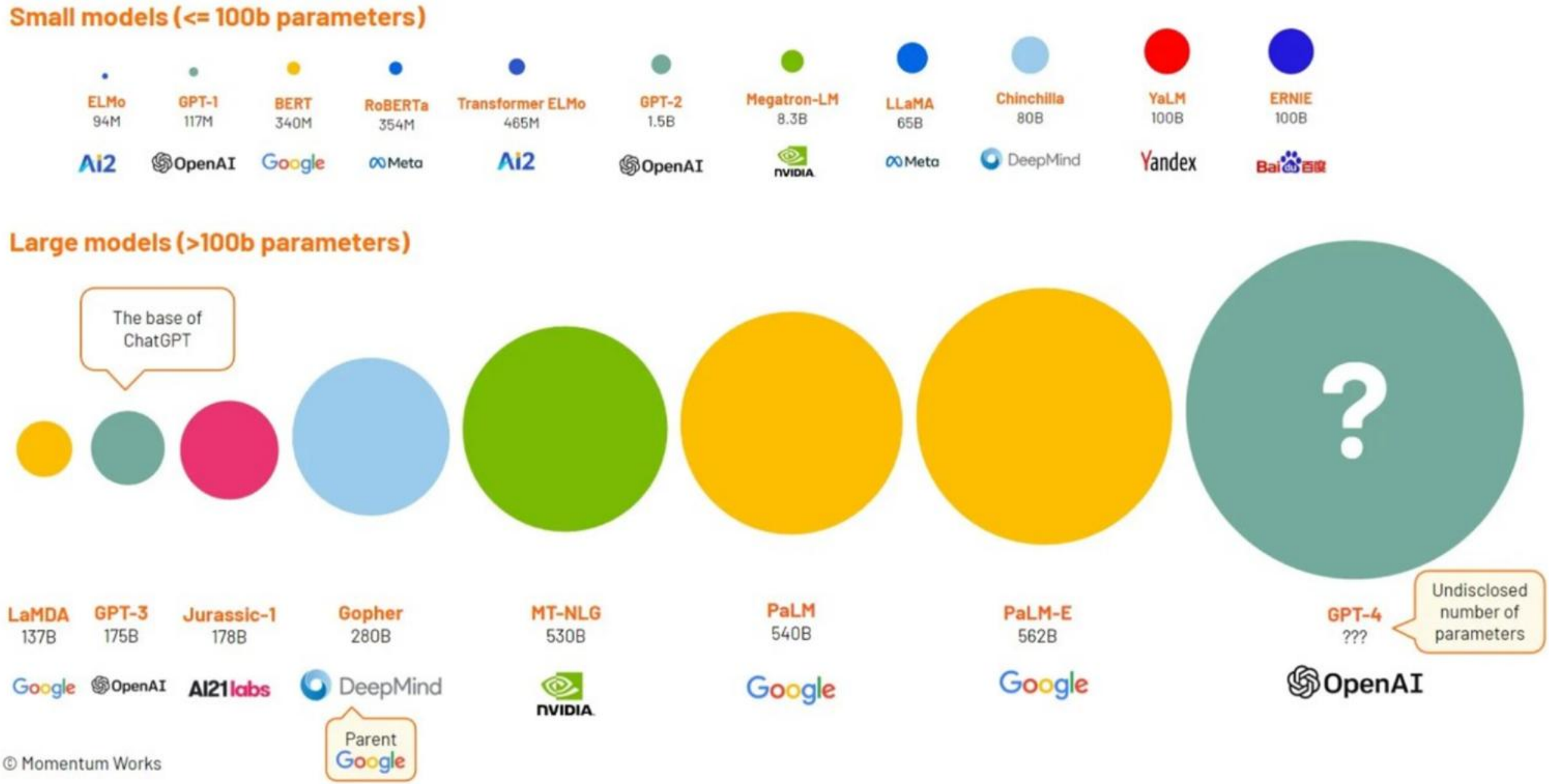


Why are LLMs called large language models?

- LLMs are called "large" because of their immense scale in both architecture and training data.
- LLMs consist of billions to trillions of trainable parameters—internal values the model learns and adjusts to identify patterns and generate accurate outputs.
- In general, the larger the model, the better its performance on a wide range of language tasks.

> For example, OpenAI's GPT-3 has 175 billion parameters, while GPT-4 is estimated to have around 1.76 trillion, demonstrating how size correlates with capability.

- In addition to their size, LLMs are trained on *massive datasets* and require *substantial computational resources*, making it impractical for most users to train them from scratch.
- This combination of vast parameter counts, massive training data, and advanced computing power is what makes these models truly large.



Source: https://thelowdown.momentum.asia/the-emergence-of-large-language-models-llms/

Image Generation - Diffusion Model

- **Definition**: Much like LLMs are the leading type of generative AI models for NLP tasks, diffusion models are the state-of-the-art approach for generating visual content like images and art.
- The principle behind diffusion models is to gradually add noise to an image and then learn to reverse this process through denoising.
- By doing so, the model learns highly intricate patterns, ultimately becoming capable of creating impressive images that often appear photorealistic.
- **Why it's key**: Diffusion models stand out in today's generative AI landscape, with tools like DALL·E and Midjourney capable of producing high-quality, creative visuals from simple text prompts. They've become especially popular in business and creative industries for content generation, design, marketing, and more.

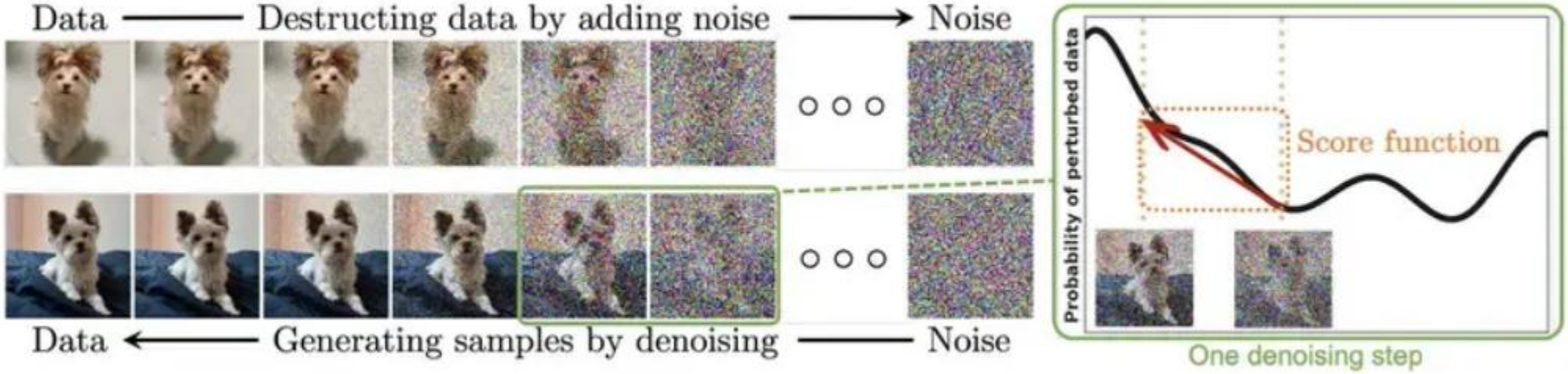


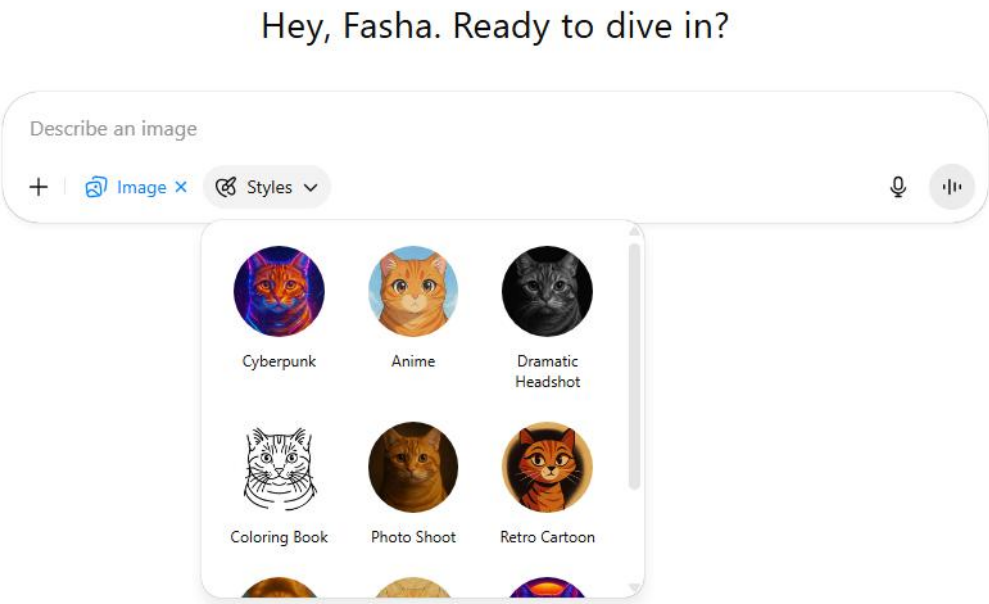
Image Generation

- Models like **DALL-E**, **Midjourney**, and **Stable Diffusion** are trained on a massive dataset of images and their corresponding text descriptions.
- When you give them a prompt like "a photorealistic image of a cat in a business suit drinking coffee," the model generates a new, original image that matches the description.
- **Business Applications:**
  - **Marketing & Advertising:** Quickly create unique visuals for social media campaigns, ad banners, or blog posts without a designer.
  - **Product Design:** Generate new product mockups or explore different design options in a fraction of the time.
  - **E-commerce:** Create lifelike product photos or virtual try-on experiences without expensive photo shoots.

1. https://openai.com/index/dall-e-2/
2. https://www.midjourney.com/home

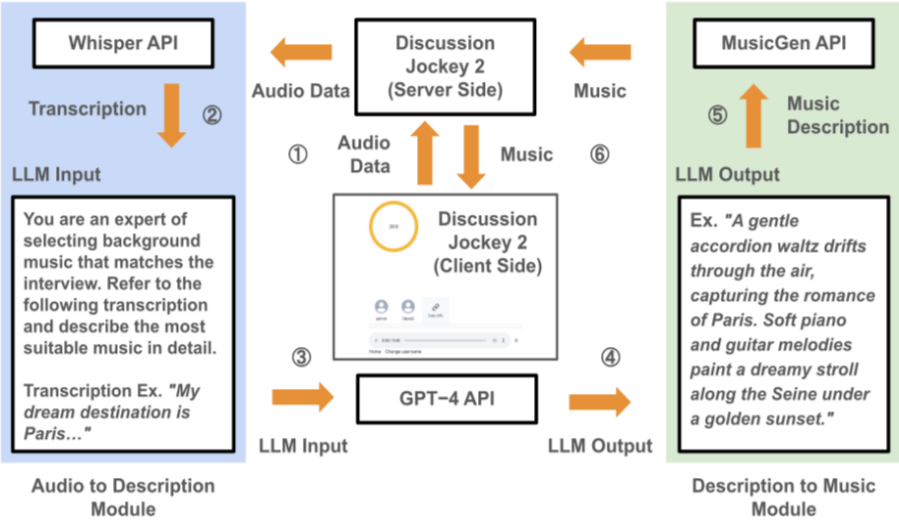
ChatGPT is a Multimodal LLM that Can Generate Images

- ChatGPT is a multimodal large language model, meaning it can understand and generate both text and images.
- You can describe an image in words, and ChatGPT will create a matching visual.
- Examples include:
  - A photorealistic cat in a business suit drinking coffee
  - A scene in the style of a coloring book, cartoon, or anime



Audio and Music Generation

- These models can create original music, sound effects, or lifelike human speech.
- They are trained on vast libraries of audio data and can generate new compositions based on a text description or a musical theme.
- **Business Applications:**
  - **Content Creation:** Generate royalty-free background music for a video or a podcast.
  - **Customer Experience:** Create natural-sounding voiceovers for virtual assistants or automated phone systems.
  - **Gaming:** Dynamically create sound effects and soundtracks that adapt to the game's environment.



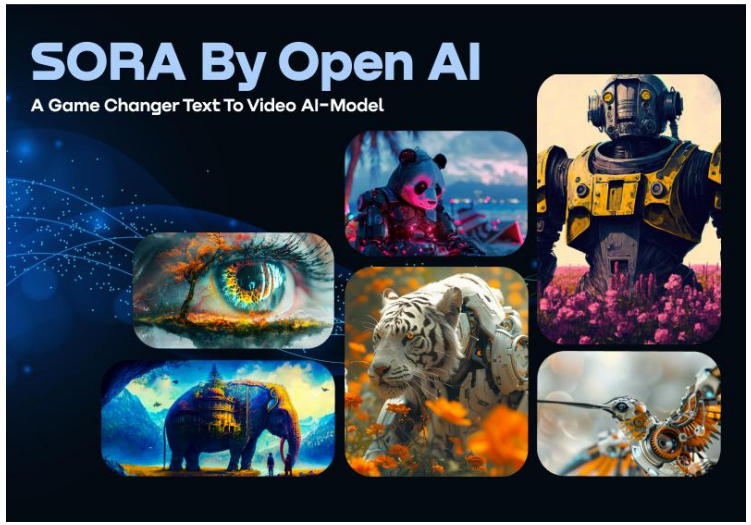
The diagram shows a workflow for automatically generating background music that matches spoken dialogue using AI models.

1. Whisper API transcribes spoken audio into text.
2. The transcription is passed into the GPT-4 API, which acts as an Audio-to-Description Module. It analyzes the context and creates a descriptive explanation of the most suitable music style.
3. This music description (e.g., “a gentle accordion waltz...”) is sent to MusicGen API, the Description-to-Music Module, which generates actual music.
4. The Discussion Jockey 2 (Server + Client Side) coordinates the flow of audio data, transcription, music description, and generated music.

Essentially, the system listens to speech, interprets the context, and then produces matching background music automatically.

Video Generation

- Models like **Sora** can be used to create high-quality, realistic video clips from simple text prompts.
- **Business Applications (Emerging):**
  - **Entertainment:** Rapidly prototype animated storyboards for films or create short marketing videos.
  - **Training:** Generate realistic simulations for employee training, such as safety or machinery operation videos.



1. https://sora.chatgpt.com/
2. https://openai.com/sora/
3. https://openart.ai/home

Limitations and Ethical Considerations

Hallucinations:

1. Definition: Generative AI models can confidently produce false, nonsensical, or made-up information.
2. Reason: They are trained to generate plausible sequences of words, not necessarily to be factual.
3. Mitigation: Always verify information generated by AI, especially for critical business decisions.

Bias:

1. Source: AI models learn from the data they are trained on. If the training data contains societal biases (e.g., gender, racial), the AI can perpetuate or amplify them.
2. Implication: AI-generated content might reflect these biases, leading to unfair or inappropriate outputs.
3. Mitigation: Be aware of potential biases and critically review AI outputs for fairness and inclusivity.

Data Privacy and Security:

1. Concern: When you input sensitive company data into public AI models, there's a risk of that data being used for training or exposed.
2. Best Practice: Avoid inputting confidential or proprietary information into general-purpose AI tools.

Copyright and Intellectual Property:

1. Issue: Who owns the content generated by AI? Can AI generate content that infringes on existing copyrights?
2. Current State: This is an evolving legal area. Be mindful of these complexities, especially for commercial use

Final Words

- The ability to generate new content is not a restricted for a specific type of data.
- It is a fundamental capability that is transforming how businesses operate across every creative and content-driven function.
- As these models become more sophisticated, the lines between different data types will blur, leading to powerful new applications.

KNIME for Generative AI

Integrate LLMs in a data analytics workflow

- You've likely used ChatGPT in a browser to ask questions.
- But how can you connect it to your own data in a way that is meaningful, scalable, and respects privacy and governance requirements?
- Manually prompting the model for every row or data point is inefficient and with large or frequently updated datasets, simply impractical.
- That's where KNIME comes in.
- **KNIME Analytics Platform** lets you build visual workflows that combine data and GenAI in one environment. With it, you can:
  - Access and process your data
  - Connect to LLMs programmatically
  - Automate prompts across rows of your data
  - Combine GenAI with classic data analytics
- In addition to KNIME Analytics Platform, helps ensure GenAI is used safely, with controls over which models can be used, how they're integrated, and what data is shared in prompts.

Getting Started

1. Download KNIME Analytics Platform

2. Install the KNIME AI Extension (Labs)—this gives you the nodes for local and remote LLM connections.

The KNIME AI extension provides dedicated nodes for connecting to LLMs and embedding models of both commercial and open-source providers; prompting and chatting with LLMs, creating and managing vector stores, as well as implementing your chatbots, RAG pipelines, and agents.

3. Choose a language model

4. Build your workflow: Select model → Prompt (via LLM Prompter or LLM Chat Prompter) → Process outputs

LLM Node Repository in KNIME

LLM Selector & Connectors

- KNIME Hub LLM Selector – lets you choose from models hosted on KNIME Hub.
- OpenAI LLM Selector – integrates OpenAI models (like GPT).
- DeepSeek LLM Selector – connects to DeepSeek models.
- Anthropic LLM Selector – integrates Anthropic's Claude models.
- IBM watsonx.ai LLM Selector – connects to IBM Watsonx models.
- HF Hub LLM Selector – pulls models from Hugging Face Hub.
- HF TGI LLM Selector – connects to Hugging Face's Text Generation Inference.
- Gemini LLM Selector – integrates Google's Gemini models.
- Azure OpenAI LLM Selector – connects via Azure-hosted OpenAI service.
- Local GPT4All LLM Selector – lets you run GPT4All models locally.

Prompting Nodes

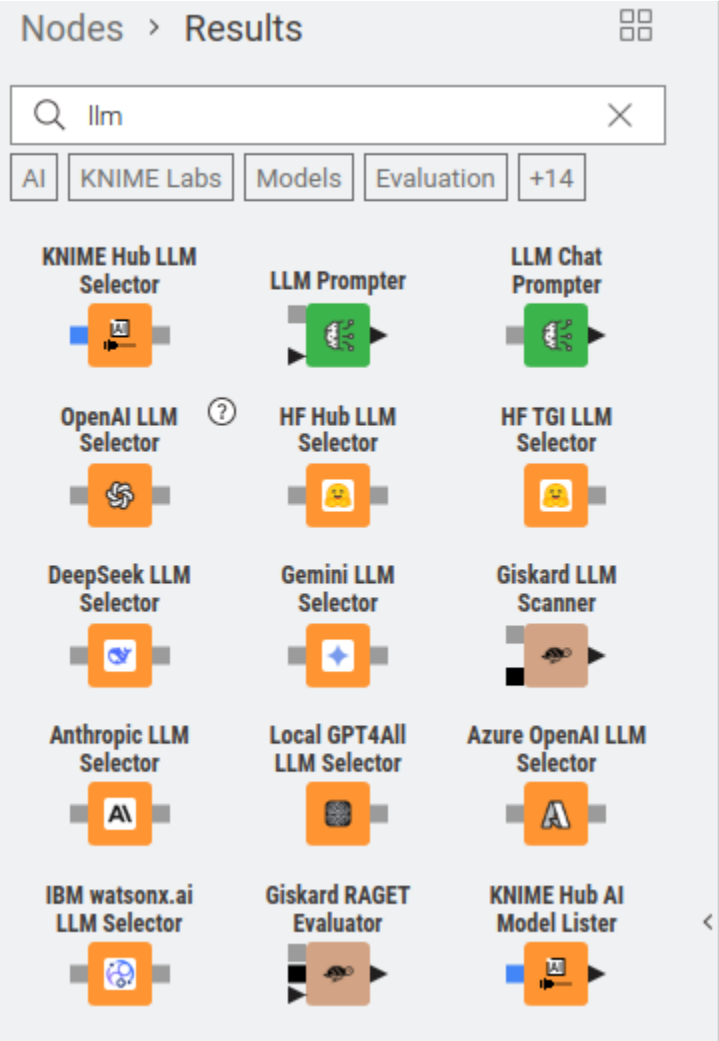
- LLM Prompter – sends prompts to a selected LLM.
- LLM Chat Prompter – tailored for conversational exchanges with chat-style models.

Evaluation & Scanning

- Giskard LLM Scanner – checks LLMs for safety, bias, robustness.
- Giskard RAGET Evaluator – evaluates Retrieval-Augmented Generation pipelines.

c

- KNIME Hub AI Model Lister – lists available AI/LLM models from KNIME Hub.



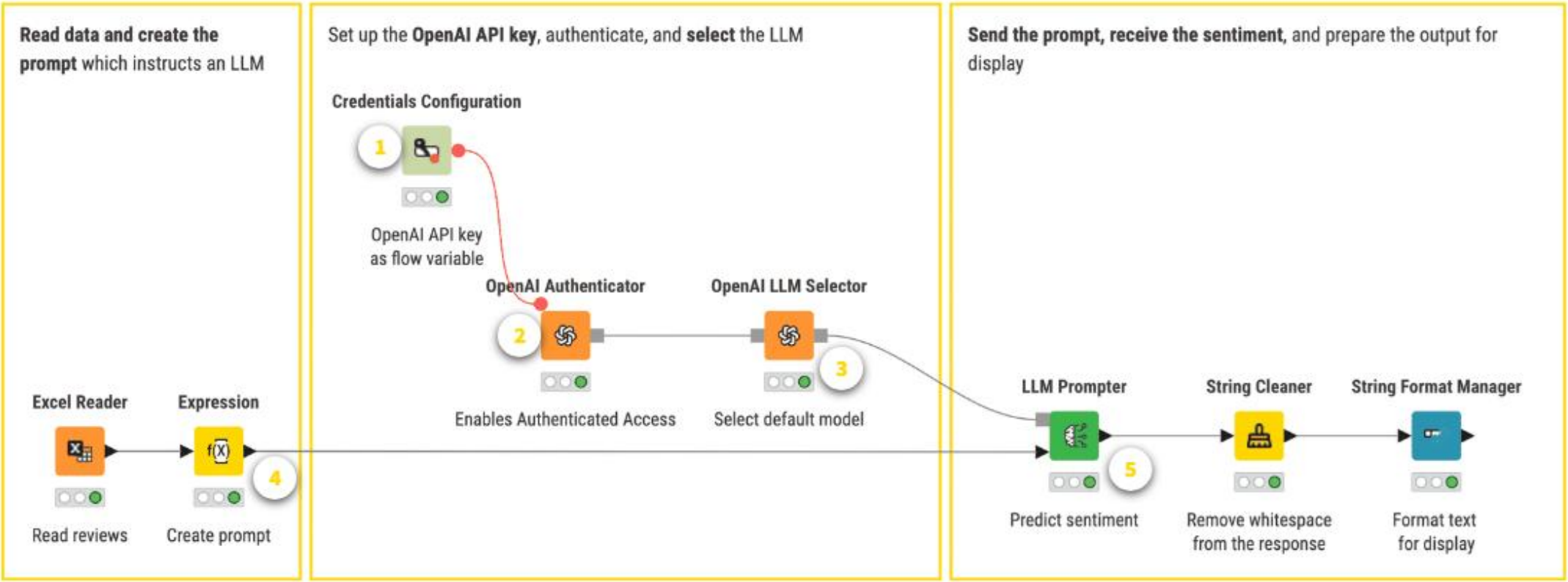
3 steps to leverage API-based LLMs

Independent of the provider, there are always 3 steps that you always need to perform:

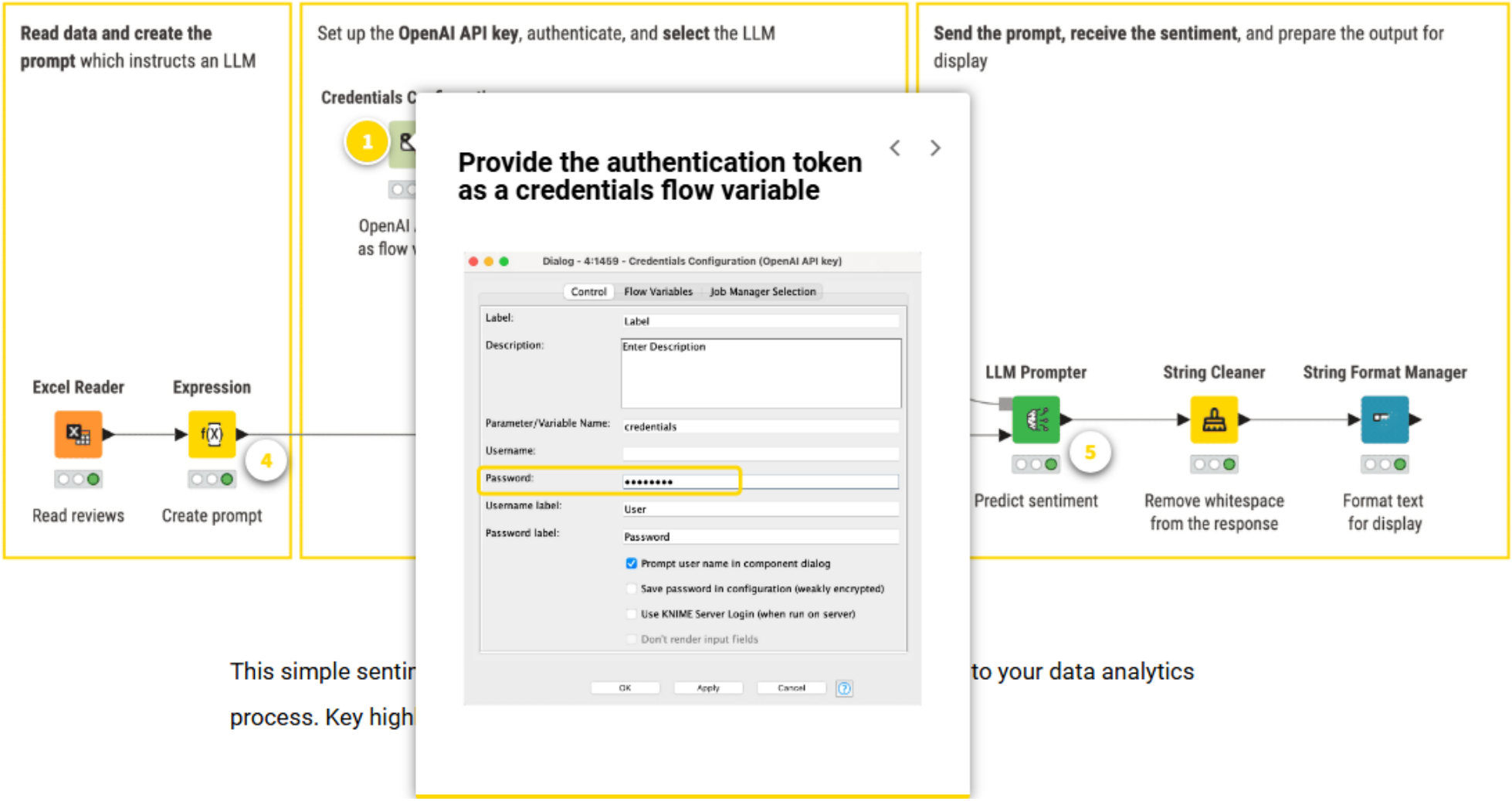
1. Authenticate against the provider & connect,
2. Select the model,
3. Prompt the model.

Sentiment Analysis Workflow

Let’s look at a simple sentiment analysis workflow, where an LLM is used to evaluate the sentiment of customer reviews.

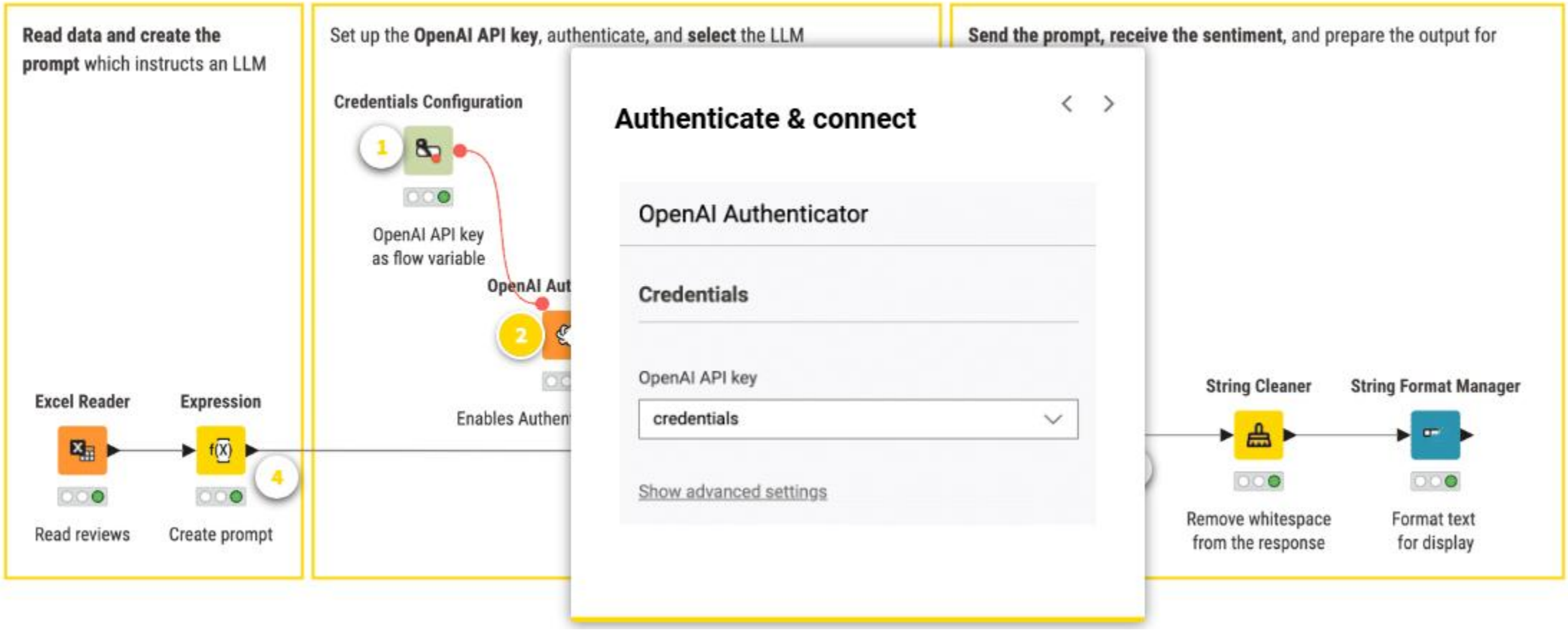


Provide Authentication Token

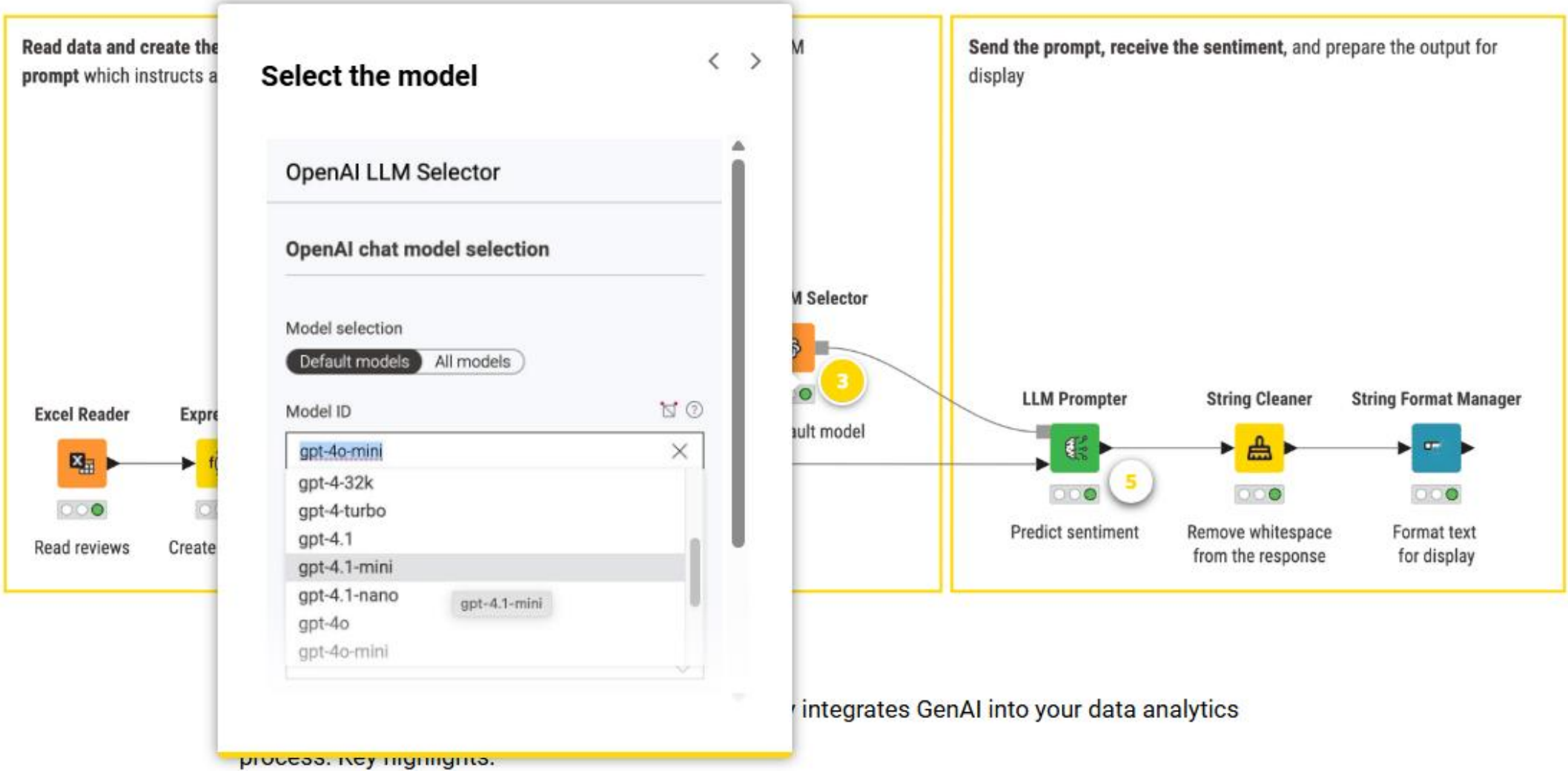


This simple sentir … to your data analytics process. Key high

Authenticate and Connect



Select the Model



Craft the Prompt

Read data and create the prompt which instructs an LLM

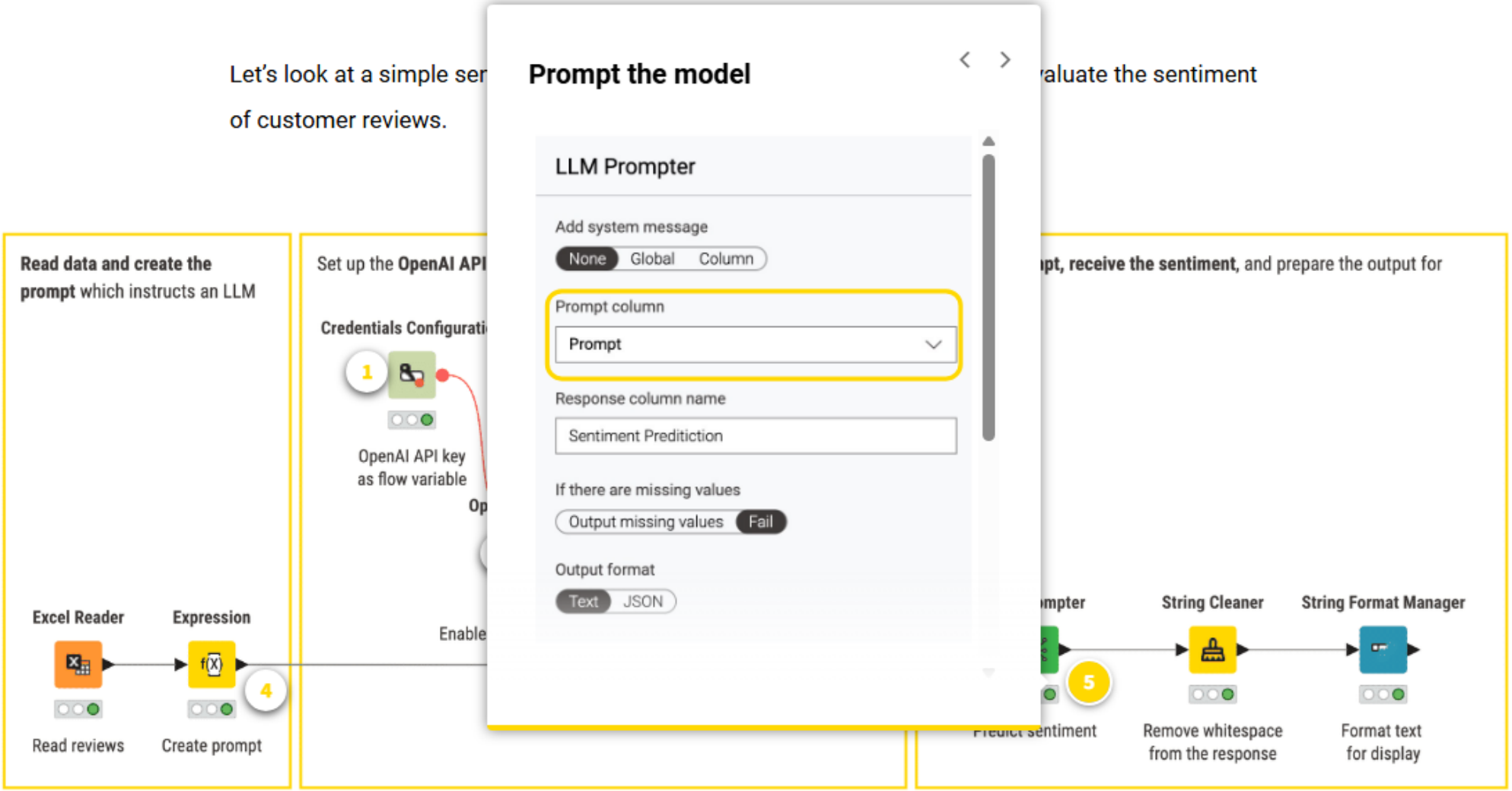
Send the prompt, receive the sentiment, and prepare the output for display

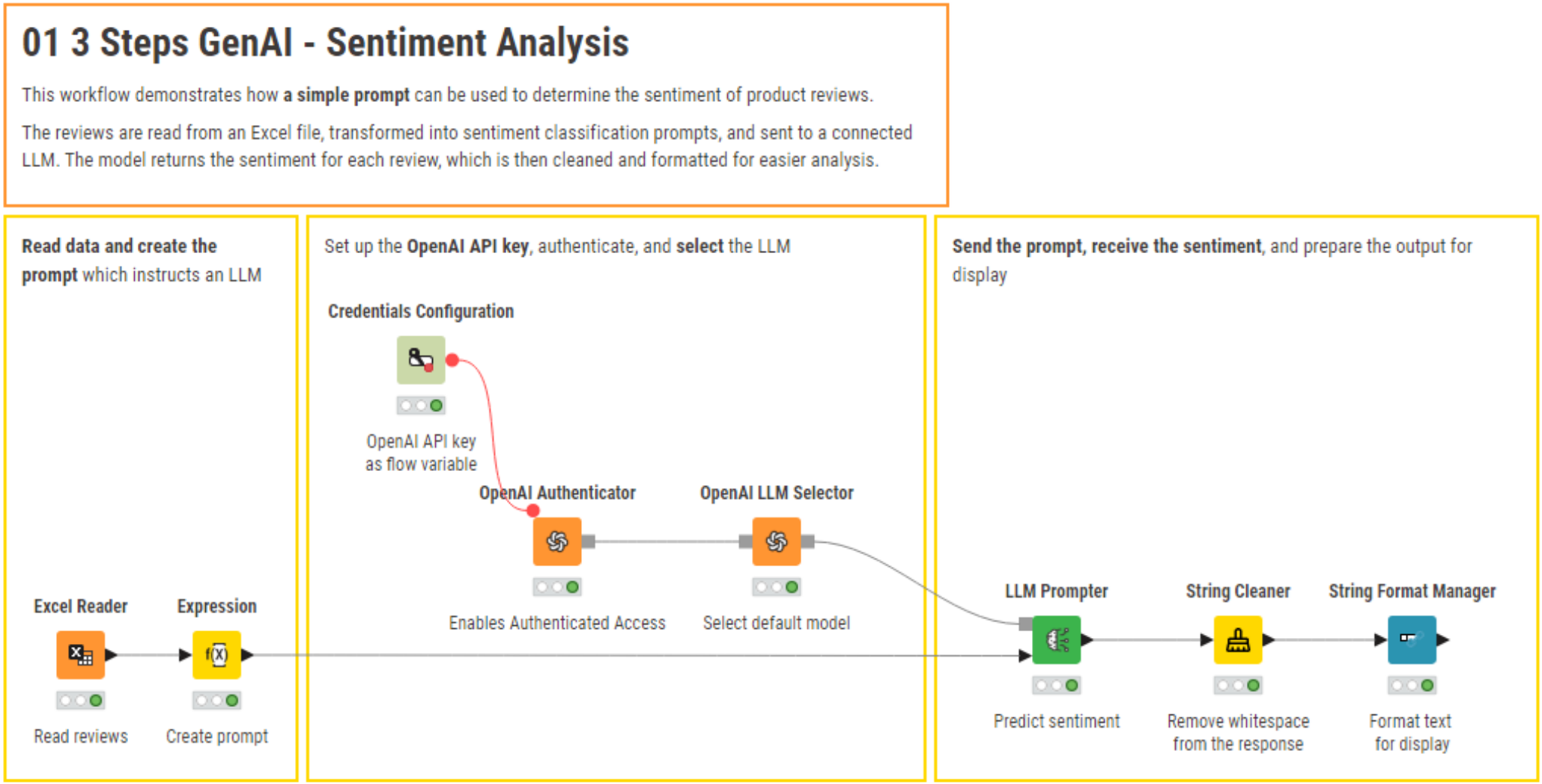
Craft the prompt

Create a new column by combining your instructions with data from other columns and flow variables, for example, using the **Expression** node.

Prompt the Model

Let's look at a simple ser aluate the sentiment of customer reviews.





The previous workflow can be downloaded as following:

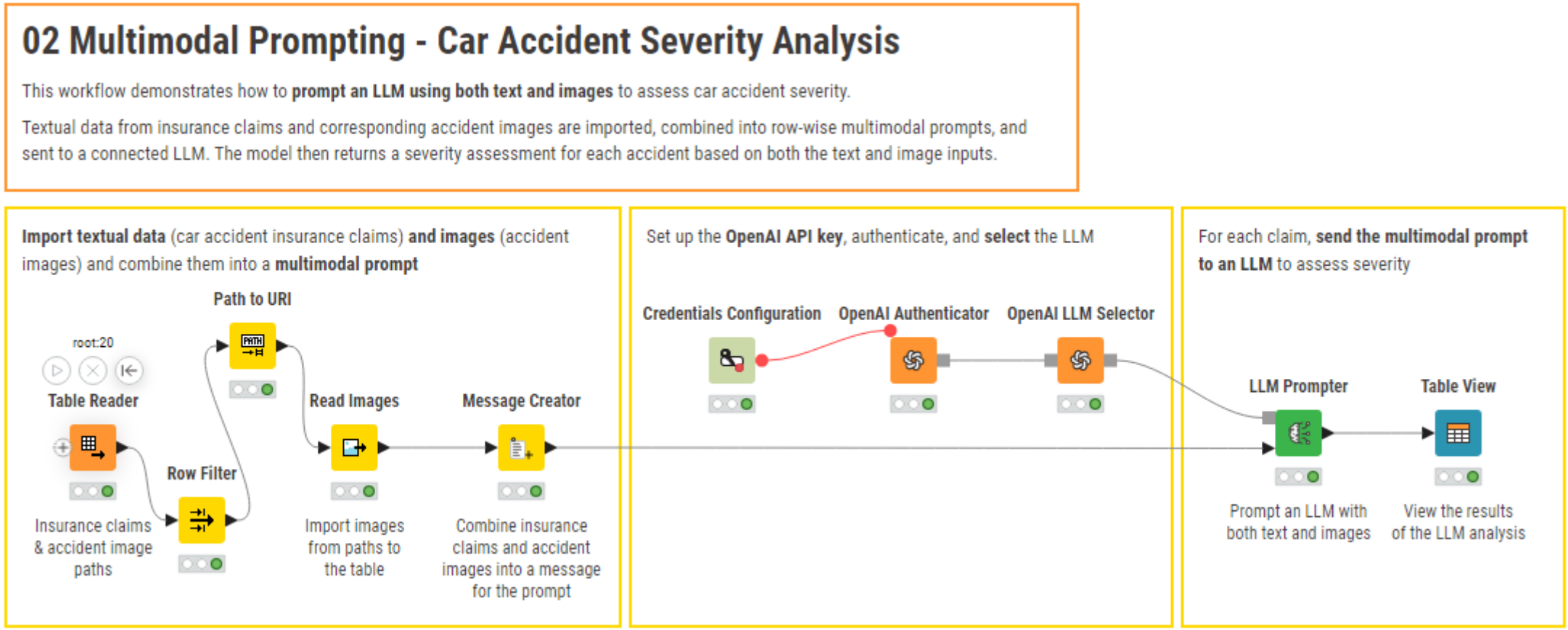
1. Download Course Workflows from VClass
2. Goto Generative AI Folder
3. Open 01 3 Steps GenAI

Sentiment Analysis Workflow

This simple sentiment analysis workflow effectively integrates GenAI into data analytics process.

Key highlights:

- **Dynamic prompting**. Prompts are automatically tailored per row by embedding data directly into them. In the example, the instruction stays the same while the review text varies row by row. You'll explore dynamic prompting in more detail in Lesson 3.
- **Row-wise processing**. The LLM analyzes review in each prompt independently and returns the response separately for each row.
- **Seamless integration**. Once sentiment is assigned by the LLM, you can proceed with classic data analysis, e.g., visualizing sentiment in a bar chart or tracking trends over time.
- The LLM Prompter processes each prompt independently, row by row, making it well-suited for GenAI-powered data analytics.



The previous workflow can be downloaded as following:

1. Download Course Workflows from VClass
2. Goto Generative AI Folder
3. Open 02 Multimodal Prompting

Local LLMs

Introduction to GPT4ALL

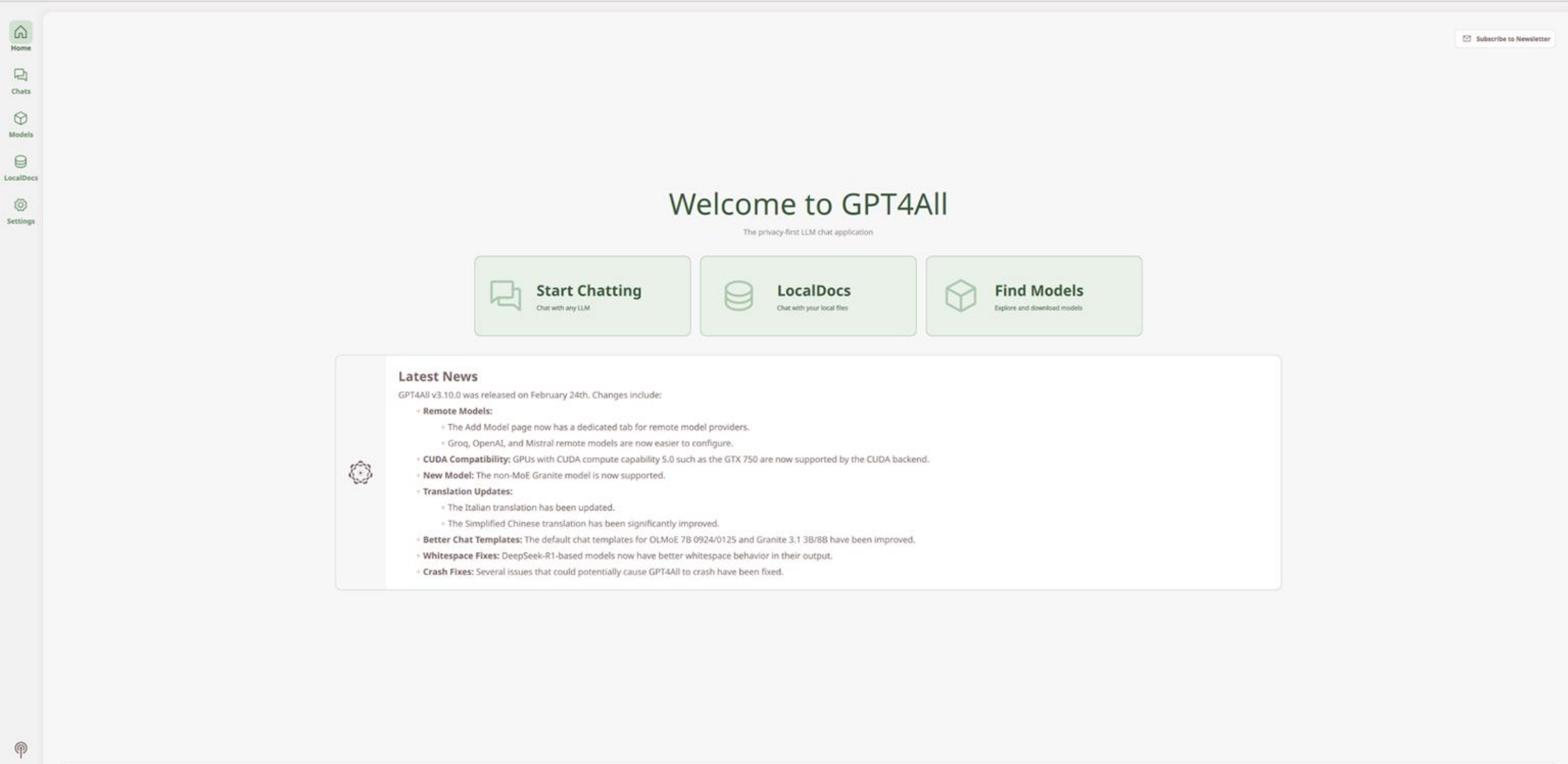
Introduction to GPT4All

- GPT4All is an open-source project from Nomic that makes it easy to run large language models entirely on your own computer.
- It bundles a cross-platform desktop chat app, a simple command-line tool, and Python bindings with a curated catalog of open-weight models.
- Those models are distributed in compact, quantized “.gguf” formats and run on lightweight runtimes (based on llama.cpp), so you can experiment on ordinary laptops without relying on cloud services or paying per token.

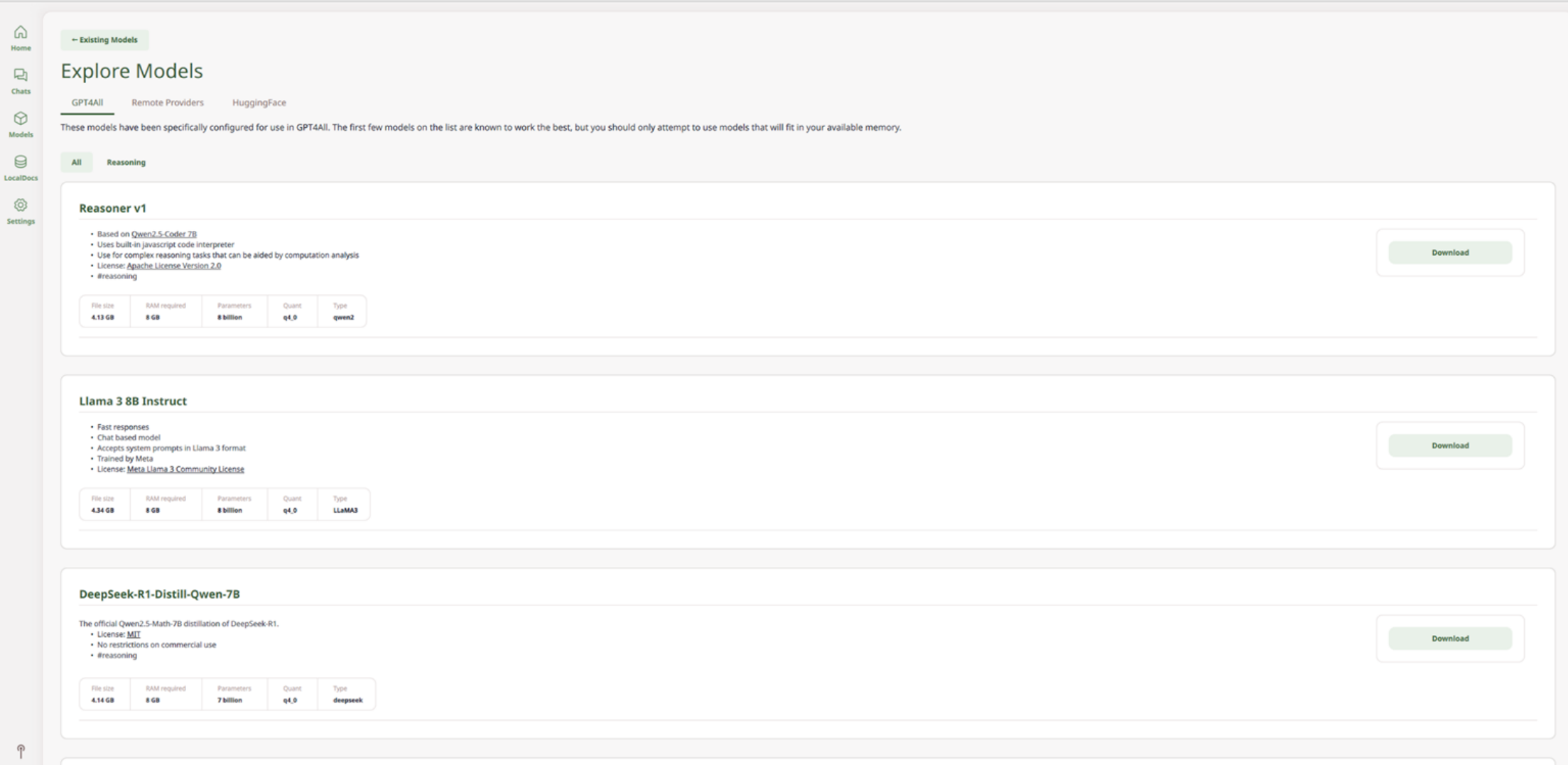
Introduction to GPT4All

- Using a **local LLM** might be a better choice:
  - **Data privacy**. Your data stays entirely within your system, which is critical when working with sensitive or confidential information.
  - **Offline availability**. Local models can run without internet access.
  - **Costs**. For organizations with high-volume usage, running models locally can reduce long-term costs, as there are no pay-per-use API fees.
- Everything runs locally, so prompts and outputs never leave your machine, there's no account or API key required, and you can work offline.
- You can choose from a range of models (instruction-tuned and chat variants of families like Llama, Mistral, Phi, and others), swap them in seconds, and tune generation settings such as temperature, top-p, context length, and max tokens.
- Performance depends on your hardware and the model size: smaller, more heavily quantized models are faster and lighter but less capable than larger ones.
- Typical uses include prototyping assistants, coding helpers, note summarizers, and RAG experiments where you combine a local model with your own documents.
- In tools like KNIME, GPT4All integrates through a local connector node, so you can drop it into a workflow, feed prompts from tables, and capture responses alongside the rest of your data pipeline.
- The main trade-offs are slower throughput than cloud GPUs, multi-gigabyte downloads for models, and quality that generally trails the newest frontier systems, but for many exploratory and privacy-sensitive tasks it offers a straightforward, zero-cost way to work with LLMs.

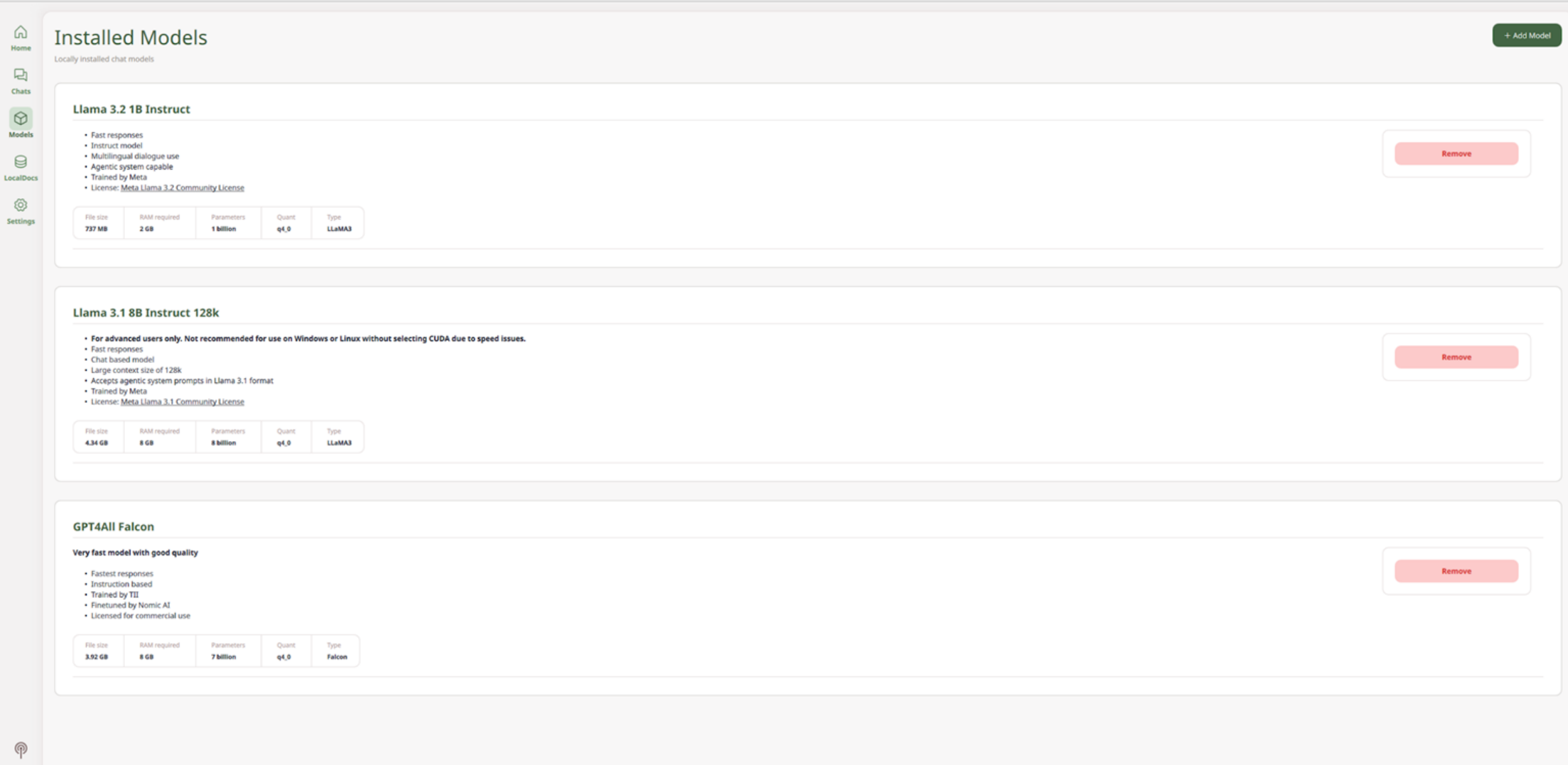
GPT4All - Home Screen



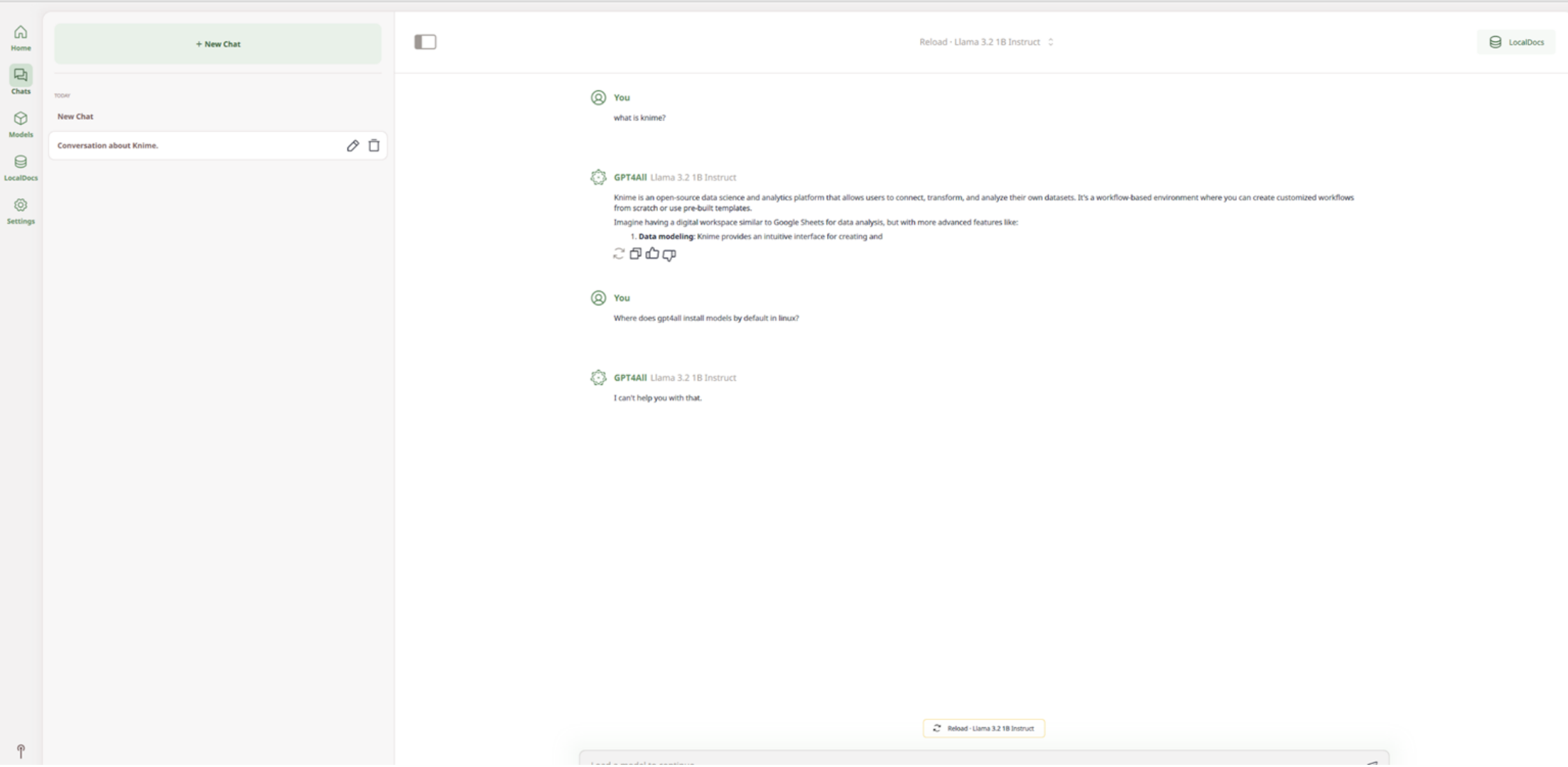
Explore and Download Models



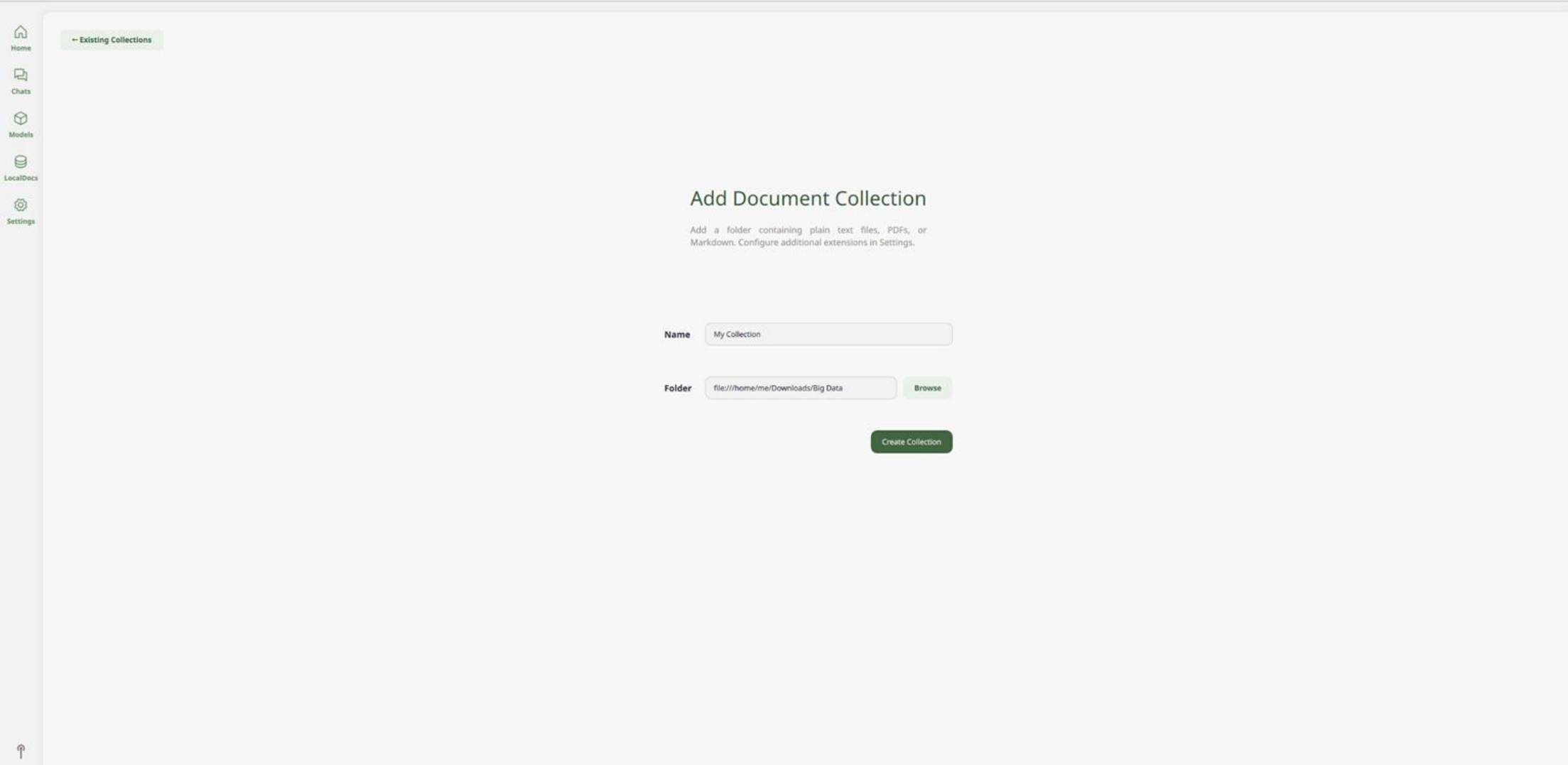
GPT4ALl – Installed Models



Chatting with Installed Models



Chat with your Own Documents



Step-by-step: KNIME with GPT4All (free, local)

Prerequisites

1. Install KNIME Analytics Platform.
2. Install the KNIME AI Extension: File → Install KNIME Extensions… → KNIME Labs → AI.
3. Ensure you have enough disk space (4–10 GB recommended) and RAM (8–16 GB recommended).

Download a GPT4All model

1. Install the GPT4All desktop app or visit the GPT4All model catalog.
2. Download a .gguf model suitable for CPU use. Choose a smaller, quantized file for speed and lower RAM, for example a Q4 or Q5 variant of an instruct model.
3. Note the full path to the downloaded .gguf file on your machine.

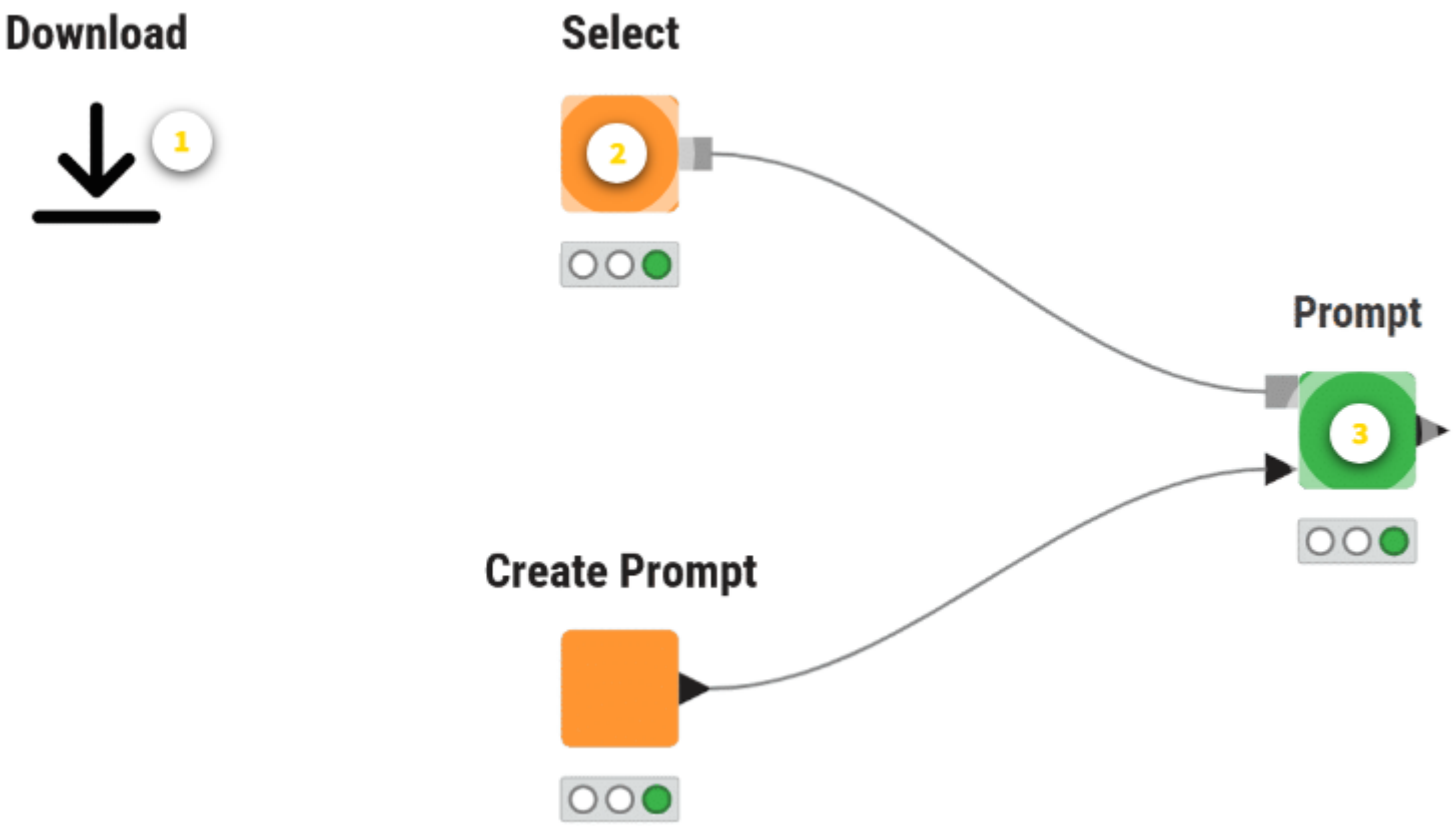
Build the KNIME workflow

1. Create a new workflow.
2. From the Node Repository, add: Local GPT4All LLM Selector, and either LLM Prompter or LLM Chat Prompter.
3. Optional: add a Table Creator or String Configuration node if you want to pass prompts from a table or a configuration dialog.

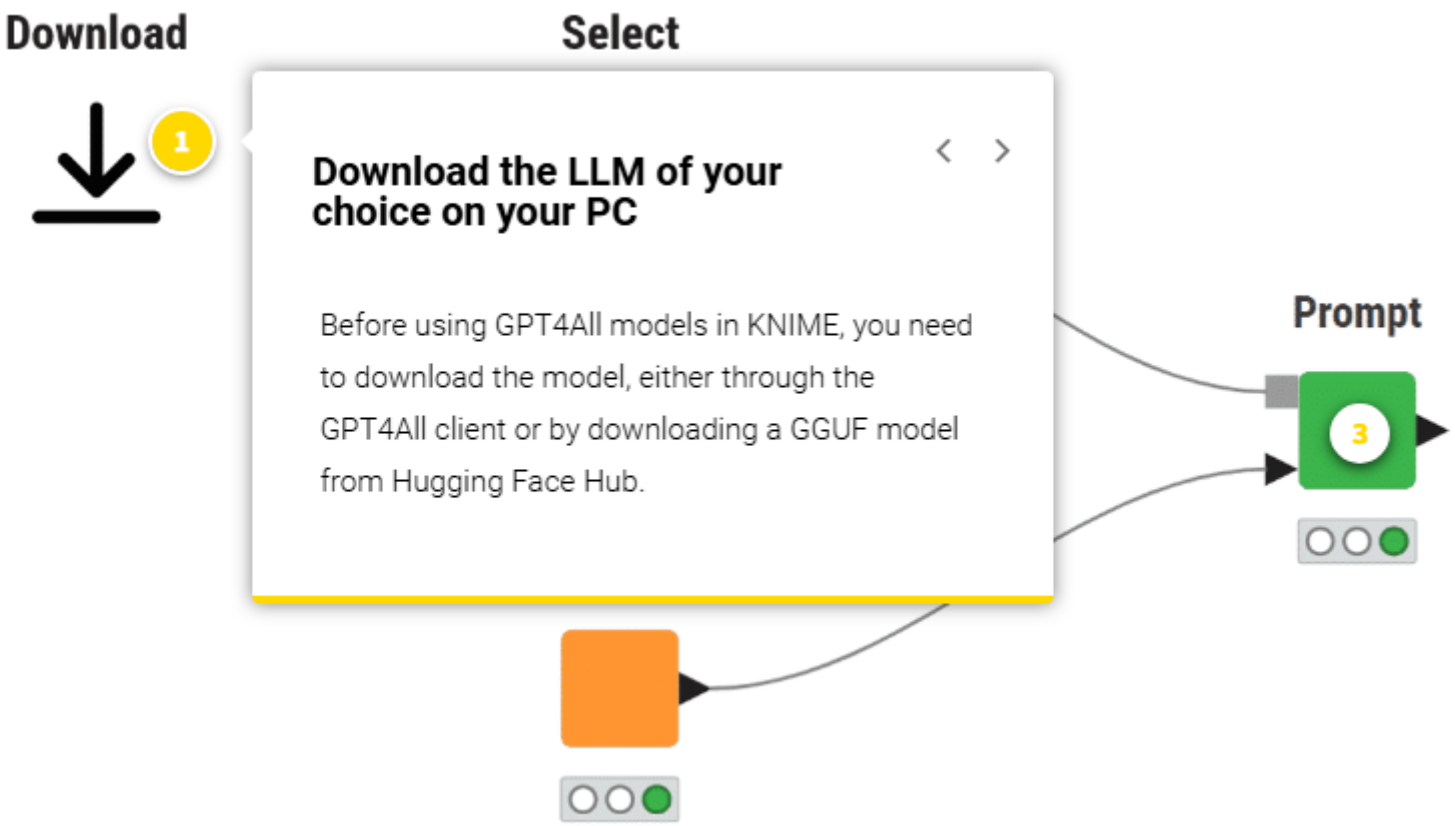
Configure Local GPT4All LLM Selector

1. Open the node configuration.
2. Model file: browse to your downloaded .gguf file.
3. Context length: start with 2048.
4. Threads: set to the number of physical CPU cores on your machine.
5. Sampling parameters: temperature 0.2–0.7, top_p 0.9 as a reasonable default.
6. Save and close.

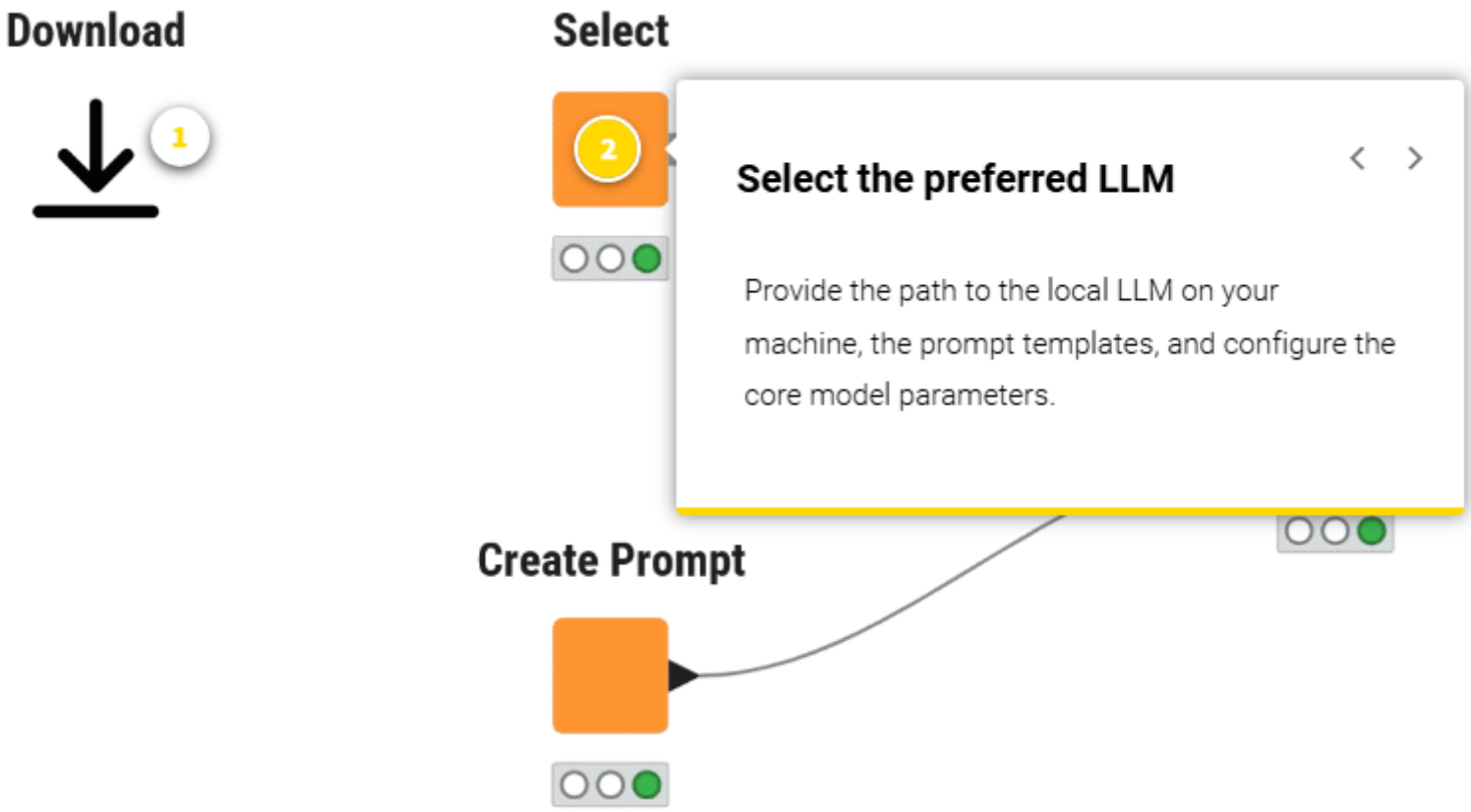
Working with Local LLMs Scenario



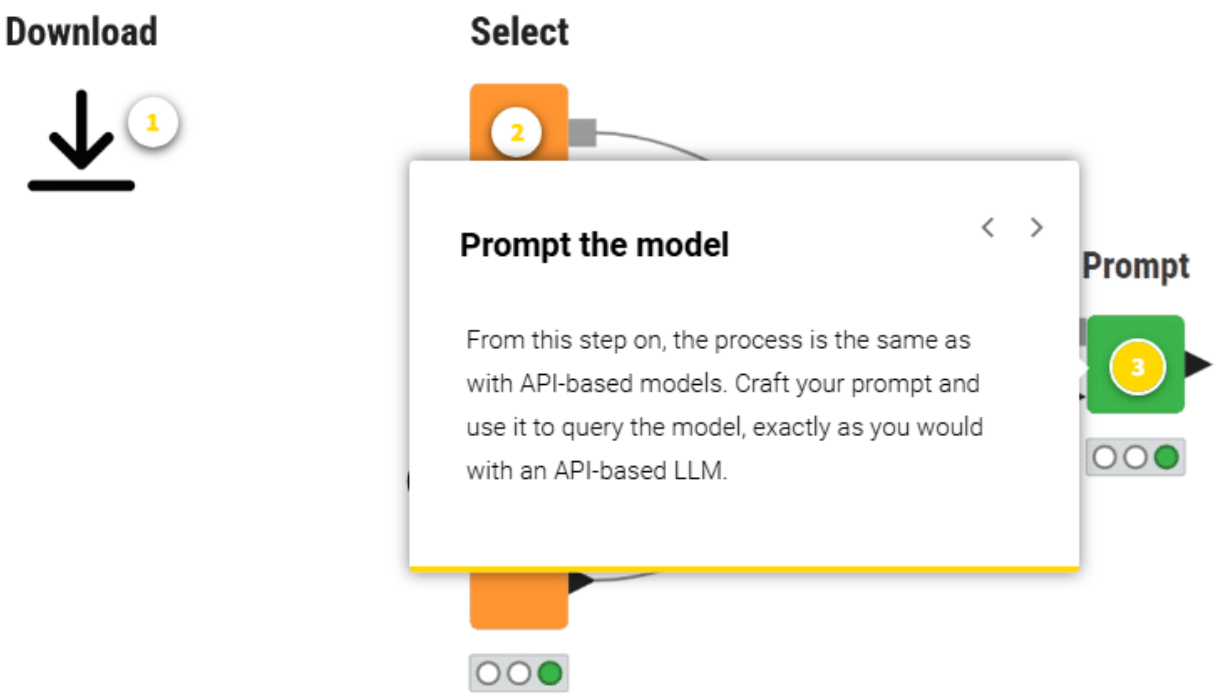
Download the LLM of your Choice



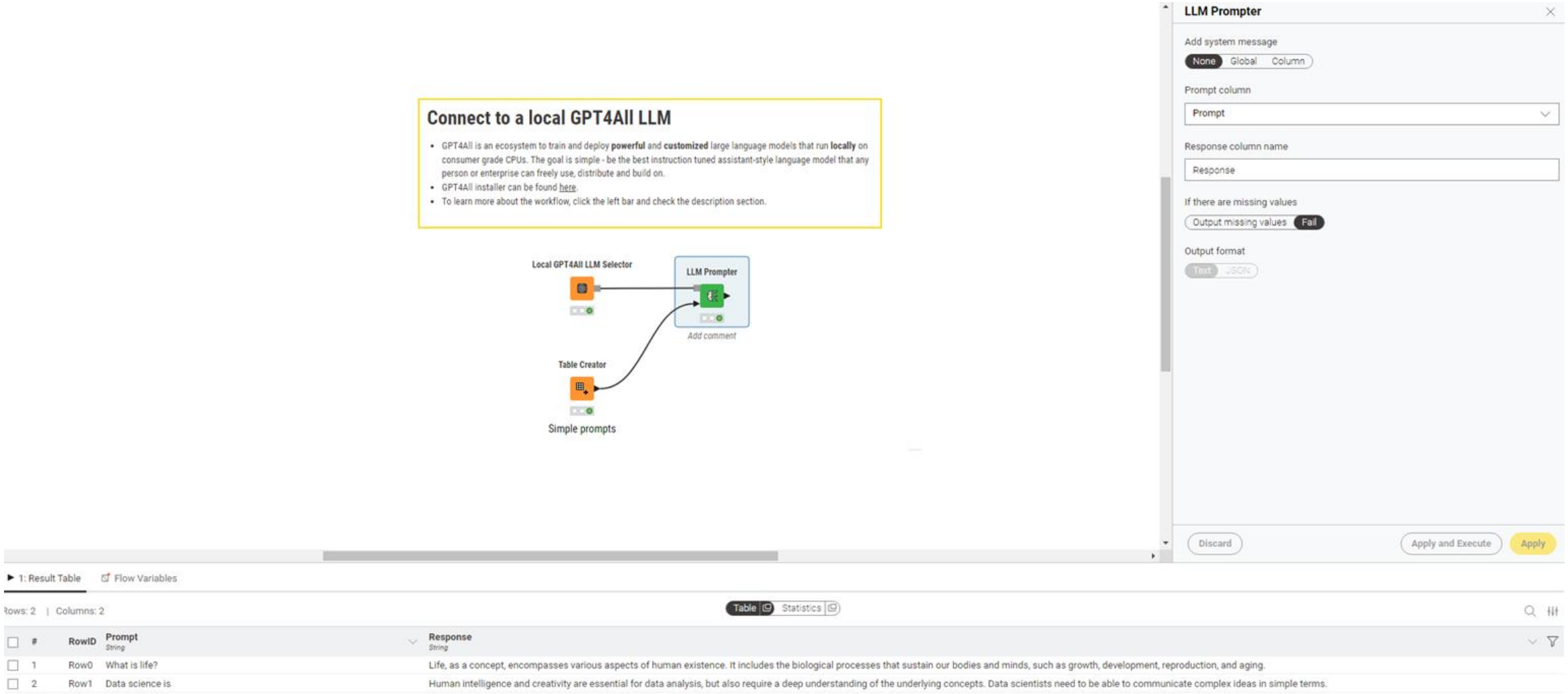
Set the Path to your LLM



Prompt the Model



You can craft your prompt as usual, without model-specific formatting. Your prompt will automatically be embedded into the template you provided in the model selector node.

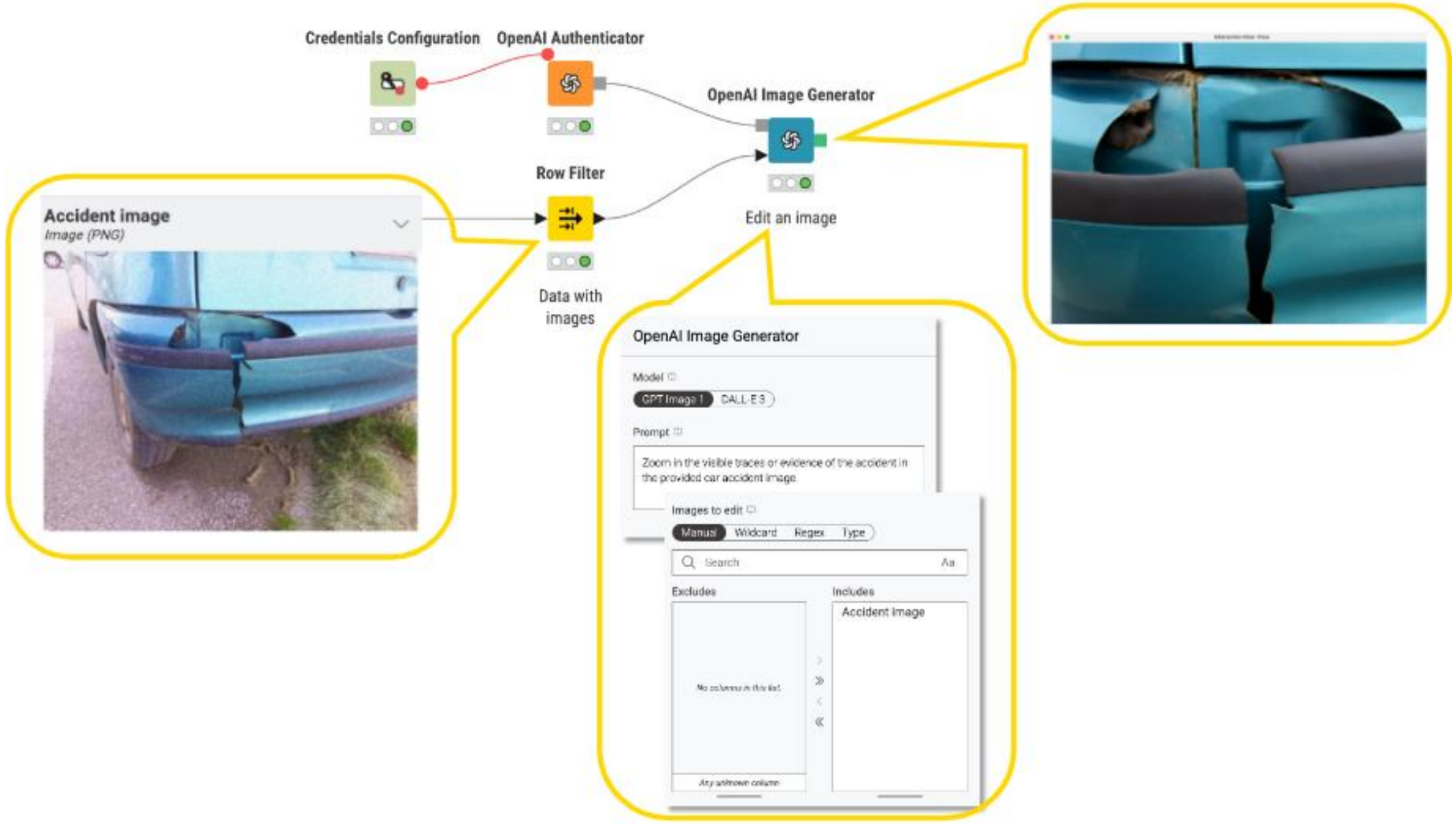


The previous workflow can be downloaded as following:

1. Download Course Workflows from VClass
2. Goto Generative AI Folder
3. Open GPT4All Workflow

Image generation

- We have learned how to prompt a model with images.
- What if we want the model to generate images in the output instead?
- Image generation can have several practical applications in data analytics:
  - **Synthetic data creation**. Generate synthetic images for analysis instead of exposing the real images.
  - **Scenario visualization**. Visualize the impact of different data-driven scenarios. For instance, how a city might look under different pollution levels or traffic models.
- In KNIME Analytics Platform, we can generate or edit images using the OpenAI Image Generator node that supports DALL-E 3 and GPT Image 1 models.
- To *generate an image*, authenticate as usual, select the model, and provide a prompt.
- The OpenAI Image Generator also offers the option to *edit images*. To do so, provide an input table through the dynamic port that contains at least one image column, provide your prompt, and select the column with images.


Credentials Configuration
OpenAI Authenticator
OpenAI Image Generator
Row Filter
Accident image
Image (PNG)
Data with images
Edit an image
OpenAI Image Generator
Model
GPT-Image 1
DALL-E 3
Prompt
Zoom in the visible traces or evidence of the accident in the provided car accident image
Images to edit
Manual
Wildcard
Regex
Type
Search
Excludes
Includes
Accident image

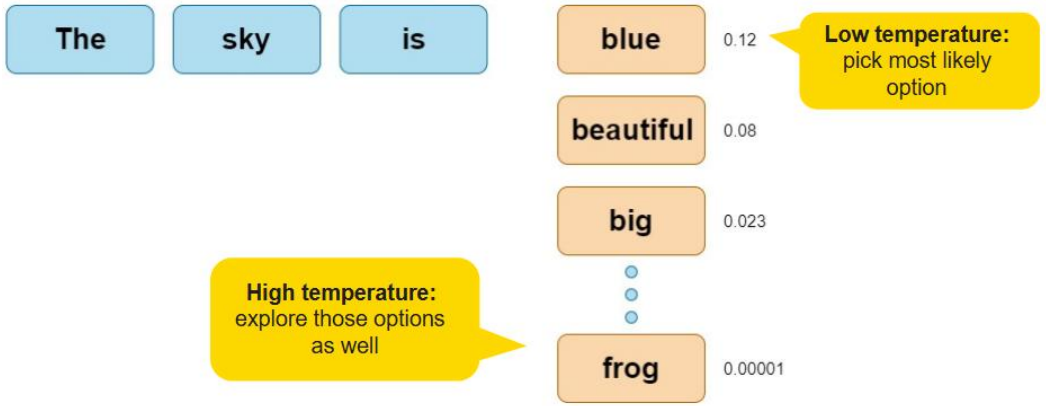
Response Control

Motivation and core model parameters

- We learned earlier in this course that LLMs can produce outputs that are varied, unpredictable, and at times inaccurate.
- To ensure consistency, comply with legal and regulatory standards, and maintain the intended tone, it’s important to guide and control these responses effectively.
- This section introduces key methods for response control, beginning with prompt engineering to shape the input, and followed by output validation techniques to refine and manage the model’s responses.
- We will explore the core parameters that we can tune to control the response:
  - Temperature and Output Length.

The Temperature

- The Temperature is a parameter that controls the randomness of an LLM's output. A lower temperature makes the model more likely to choose the highest-probability token, resulting in more deterministic responses. A higher temperature increases the likelihood of selecting less probable tokens, leading to less deterministic and more creative outputs.
- The temperature typically ranges from 0.0 - 1.0, though some implementations allow values up to 2.0 to produce extremely random and creative outputs. The temperature gives you a way to control the balance between precision and creativity, depending on the task at hand, without changing the underlying model itself.



Context Length and Tokens

- Another important concept for response control is the context length.
- A context length, as well as prompt and output length are usually measured in tokens.

What is a token?

- A Token is one input or output unit.
- A token might be a full word, part of a word, or even a single character, depending on the model.

Example Tokenizers:

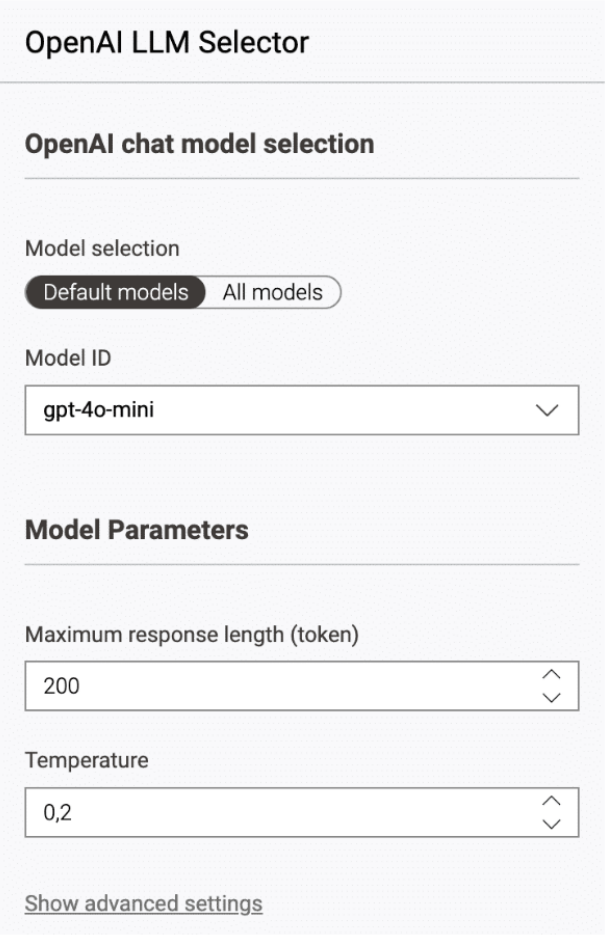
  - https://platform.openai.com/tokenizer
  - https://huggingface.co/spaces/Xenova/the-tokenizer-playground

Context length

- Maximum number of tokens that can be processed by an LLM.

Context Length

- While we can’t control the context length—it’s a fixed parameter determined by the chosen LLM—it’s important to be aware of it when designing prompts, especially for tasks involving long documents or extended conversations.
- What we can control is the prompt length and the response length.
- Specify the desired response length (in token) and include clear instructions in the prompt itself.
- If the context window is too short and the model isn’t guided to produce concise outputs, responses may be cut off mid-generation.
- You can control both temperature and output length in the configuration of an LLM Selector node of your choice.
- Tuning core model parameters like temperature and response length is important, but it is only the first step. The way you craft the prompt, known as prompt engineering, is as crucial for effective response control.



Prompt Engineering

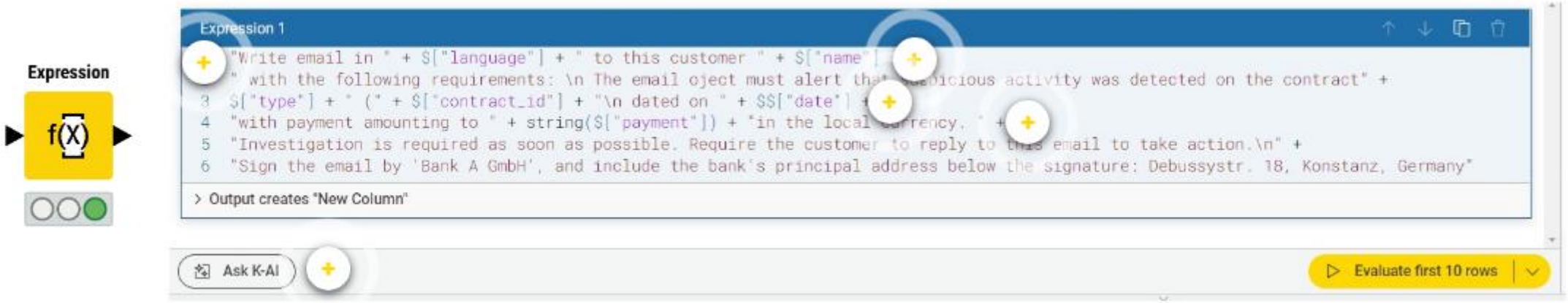
- Since an LLM doesn't "know" anything in the human sense, we have to tell it exactly what you want it to do. This instruction is called a **prompt**.
- **What is a prompt?** A prompt is the text you provide to a generative AI model to guide its output. It can be a question, a statement, or a series of instructions. Think of it as your primary way of interacting with the AI.
- **Prompt Engineering:** The better your prompt, the better the output. The art and science of writing effective prompts is called **prompt engineering**. It's the most valuable skill for a business user of generative AI. Good prompts are:
  - **Specific:** Don't just say "Write an email." Instead, say "Write a formal email to a client explaining a project delay.“
  - **Provide Context**: Give the AI necessary background information (e.g., audience, purpose, tone).
  - **Persona:** Give the AI a persona to adopt. For example, "Act as an experienced marketing analyst and provide three key takeaways from the sales data below."
  - **Structured:** Define the format you want. "Provide the answer as a bulleted list with a maximum of three points.“, "Write a 2-paragraph email“.
  - **Iterate and Refine**: Prompts are rarely perfect on the first try. Experiment!

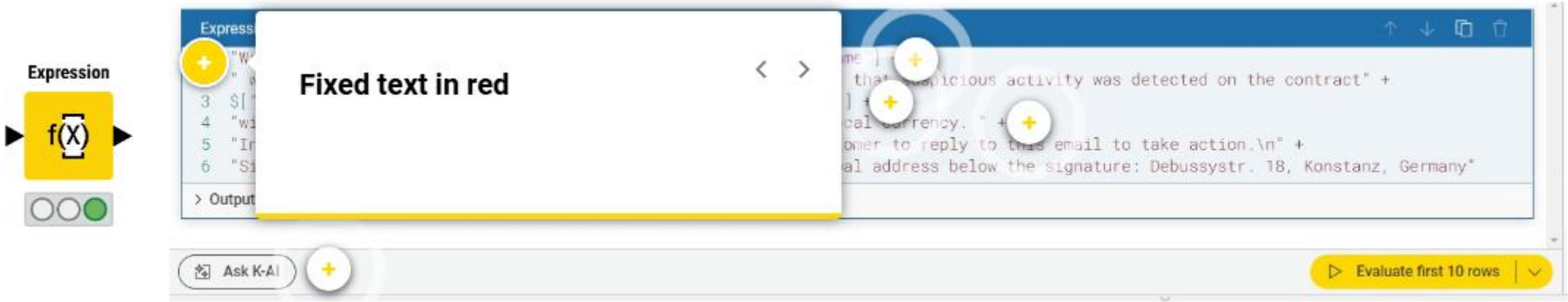
Prompt engineering

- Explore the key guidelines for effective prompts in the image below.

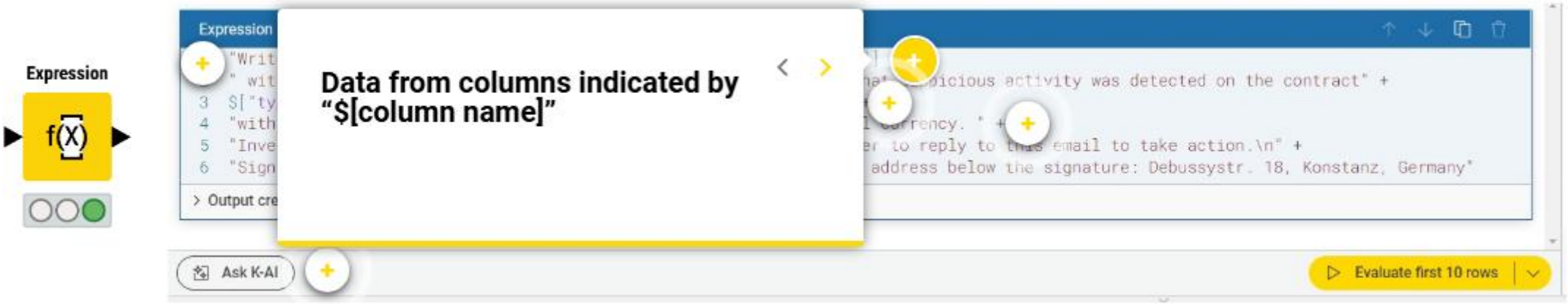
Dynamic Prompt

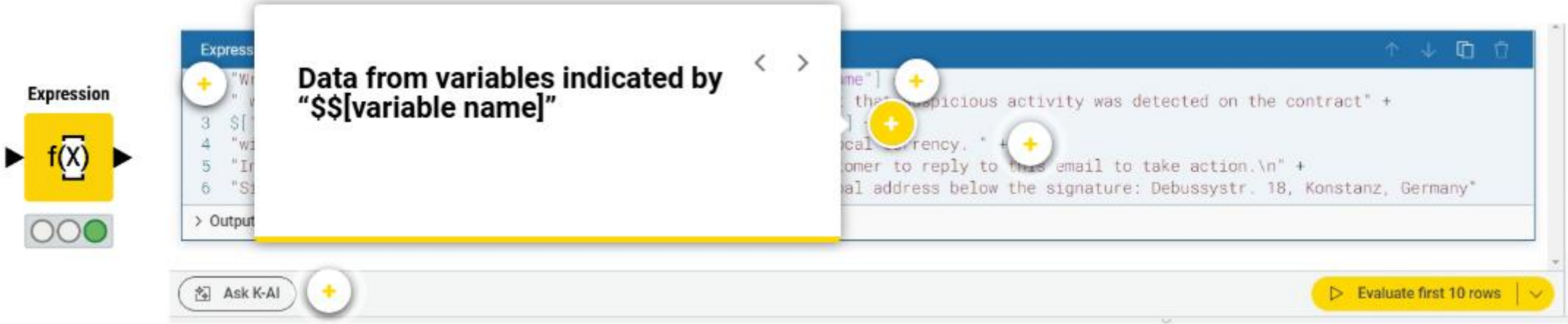
- While manually writing a few prompts might be manageable, this approach quickly becomes inefficient and error-prone when you need to generate responses for hundreds or even thousands of inputs.
- To address this, we can create a prompt template where key parts of the input are filled in dynamically using data from tables (columns) and variables.
- This approach not only saves time but also ensures consistency across prompts and allows your workflow to scale efficiently with larger datasets.
- In KNIME Analytics Platform, we can use the Expression node to construct dynamic prompts by combining fixed text with your data from tables and variables.
- We can then pass automatically and dynamically adjusted prompts to the LLM for scalable interaction.

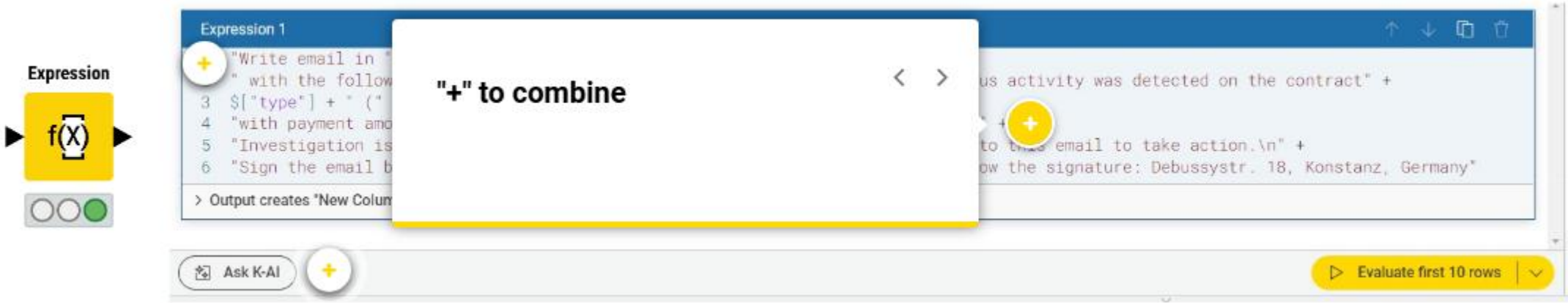


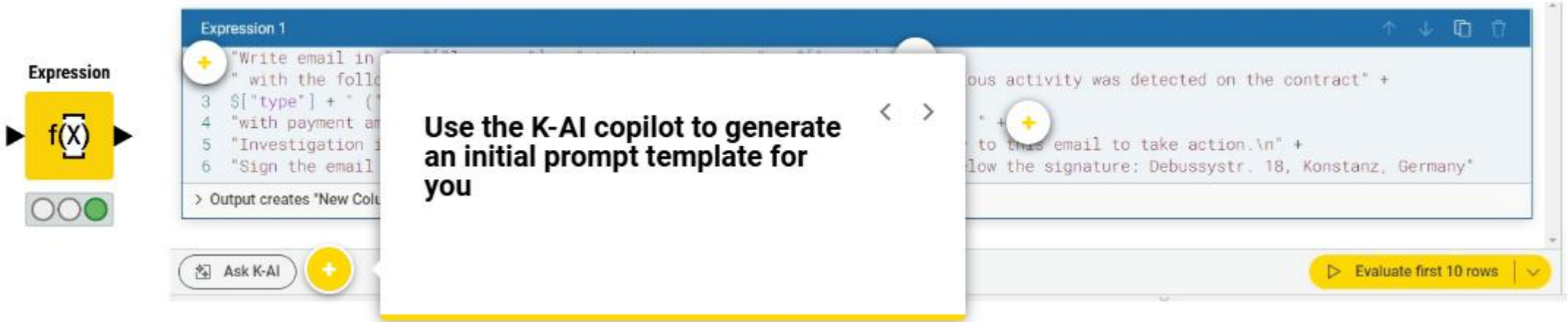


<u>Dynamic prompts</u> are essential for integrating your data into GenAI-powered analytics, as they allow you to combine fixed instructions with values from columns and variables!


Expression
f(X)
Expression
Data from columns indicated by "$[column name]"
Ask K-AI
Evaluate first 10 rows


Expression
f(X)
Data from variables indicated by "$$[variable name]"
Ask K-AI
Evaluate first 10 rows





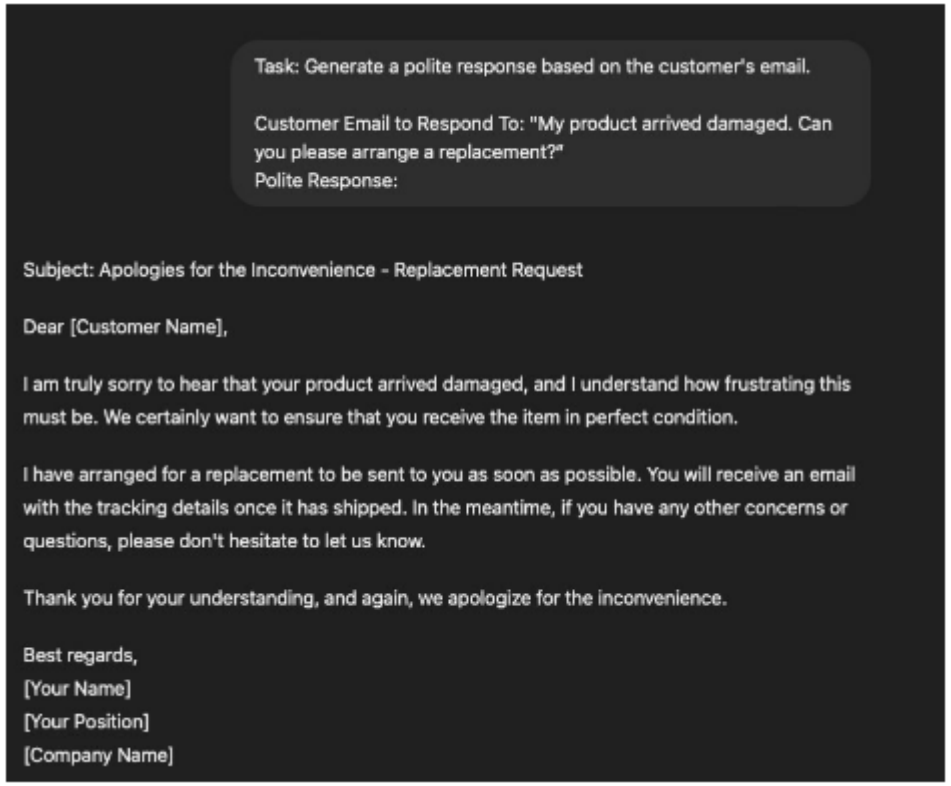
The Message Creator node can take our dynamic prompts a step further by enabling you to enrich prompts with both text and images drawn directly from your data.

In-context learning

- While dynamic prompts are essential, we will explore some specialized techniques to improve the accuracy of model responses.
- In-context learning refers to an LLM’s temporary ability to learn how to perform a specific task based on examples provided within the prompt, without requiring additional training.
- This can be done through one-shot or few-shot learning, where one or more examples are included directly in the prompt to demonstrate how the task should be completed.
- When no examples are provided, the approach is known as zero-shot prompting.

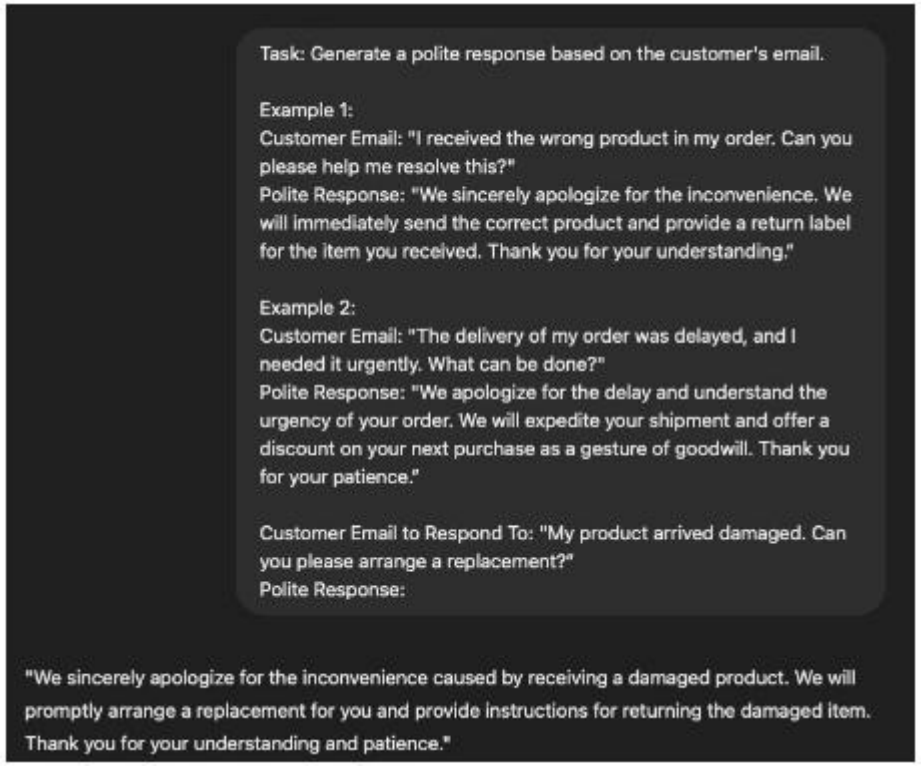
Zero Shot Learning

- In this approach, the LLM is instructed to generate a polite email response based on the customer’s request and no examples are provided.
- While the response is polite as requested, it may not fully align with the company’s preferred tone.
- For instance, if our brand is known for brief, casual communication, this response might be too formal.
- Providing examples could help achieve the desired result.
-



Few Shots Learning

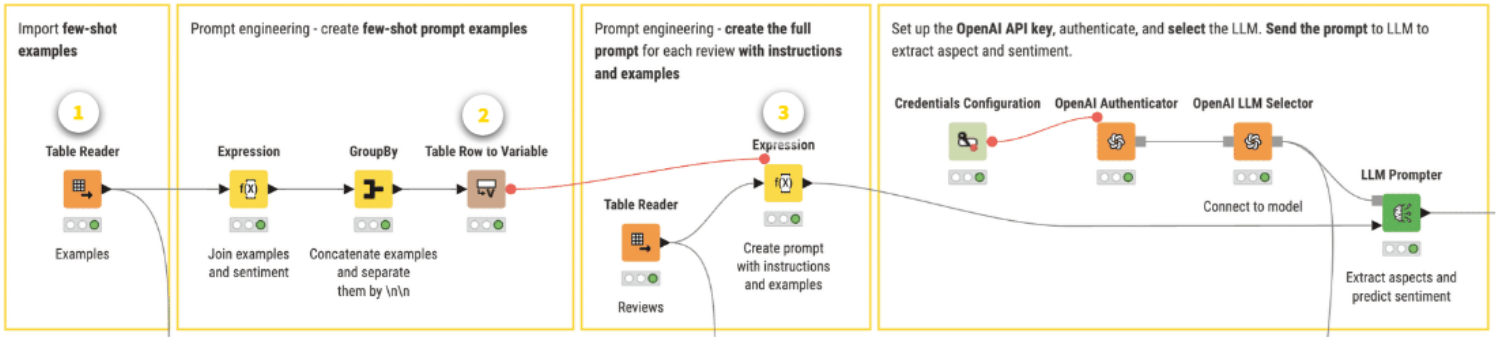
- In this approach, the LLM uses few-shot learning to generate responses by analyzing patterns from prior examples given in the prompt.
- This allows the model to pick up on patterns, such as acknowledging the issue, offering a solution, and closing with a polite note of gratitude, closely aligning with the examples provided.



Limitations

- While in-context learning can be a powerful and convenient way to guide model behavior it also comes with some limitations that are important to keep in mind.

Benefits	Limitations
• Provide examples of requirements that are hard to describe (e.g. style or tone). • Automatically adjust response by changing the examples.	• As the number of examples shown increases, the cost of inference and latency also rise. • With an increasing number of examples, it's common for the LLM to disregard some of them. • LLMs might reproduce the knowledge presented in the examples.

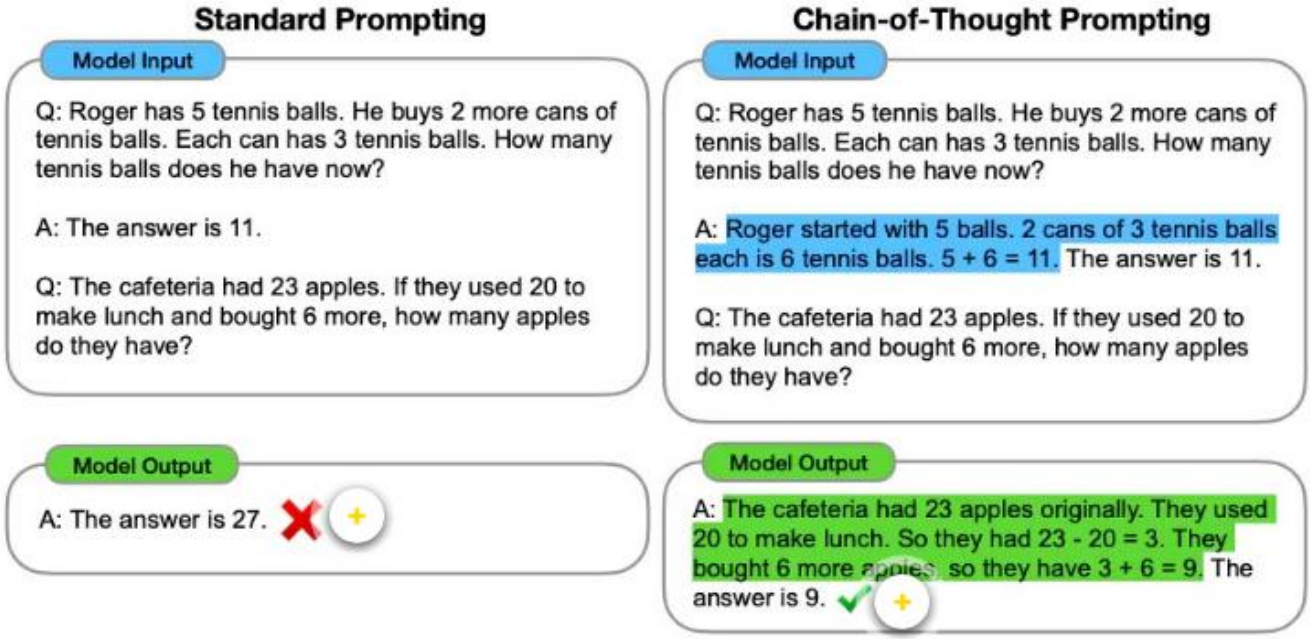


Chain of thought (CoT)

- Up to this point, we have explored the fundamentals of prompt engineering and learned about in-context learning.
- Another powerful technique is Chain of Thought (CoT) prompting.
- **Chain of Thought** Is a method that encourages an LLM to reason step-by-step by including prompts or examples that demonstrate explicit logical thinking or multi-step reasoning, leading to more accurate and explainable outputs.
- Chain of Thought prompting is particularly useful for tasks that require logical reasoning or multi-step problem-solving.
- It works by encouraging the model to reason step by step, rather than jumping directly to a final answer.
- This often results in outputs that are not only more accurate but also easier to understand and evaluate.

Chain of thought (CoT)

- To apply CoT prompting, you can include examples that demonstrate intermediate reasoning steps for a similar, already-solved task. Alternatively, you can guide the model by providing step-by-step instructions or asking structured questions that break the problem into manageable parts.
- Let’s explore an example to see how this works in practice.



Prompt formatting

- So far, we learned about prompt content.
- Another important aspect of creating effective prompts is prompt formatting.
- Up to this point, we have mostly used plain text, but adding structure and formatting to our prompt can provide the model with additional semantic cues that help guide its response more precisely.
- A practical and efficient way to do this is by using Markdown.
- Markdown allows you to format your prompts using elements such as headings, bold or italic text, underlining, lists, tables, and code blocks.

Prompt formatting

```
### **Task:**
**Classify the following sentence as either "positive"
or "negative" sentiment.**
---
#### **Example:**
- **Sentence:**
  "I love this product! It works perfectly."

- **Answer:**
  Positive
---
### **Sentence to Classify:**
"This service was disappointing."

### **Answer:**
```

Task:

Classify the following sentence as either "positive" or "negative" sentiment.

Example:

- **Sentence:**
  "I love this product! It works perfectly."
- **Answer:**
  Positive

Sentence to Classify:

"This service was disappointing."

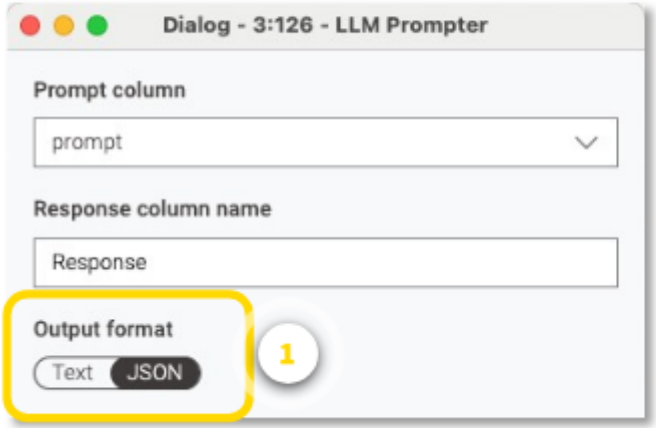
Answer:

- This kind of formatting helps highlight key information and improves the clarity of the prompt.
- Markdown is especially useful because it is more concise than HTML. Most LLM providers charge based on the number of tokens or characters used, and since Markdown uses fewer characters and is natively supported by OpenAI models, it is a convenient and effective choice for prompt formatting.

Output format and validation

Structured output and JSON mode

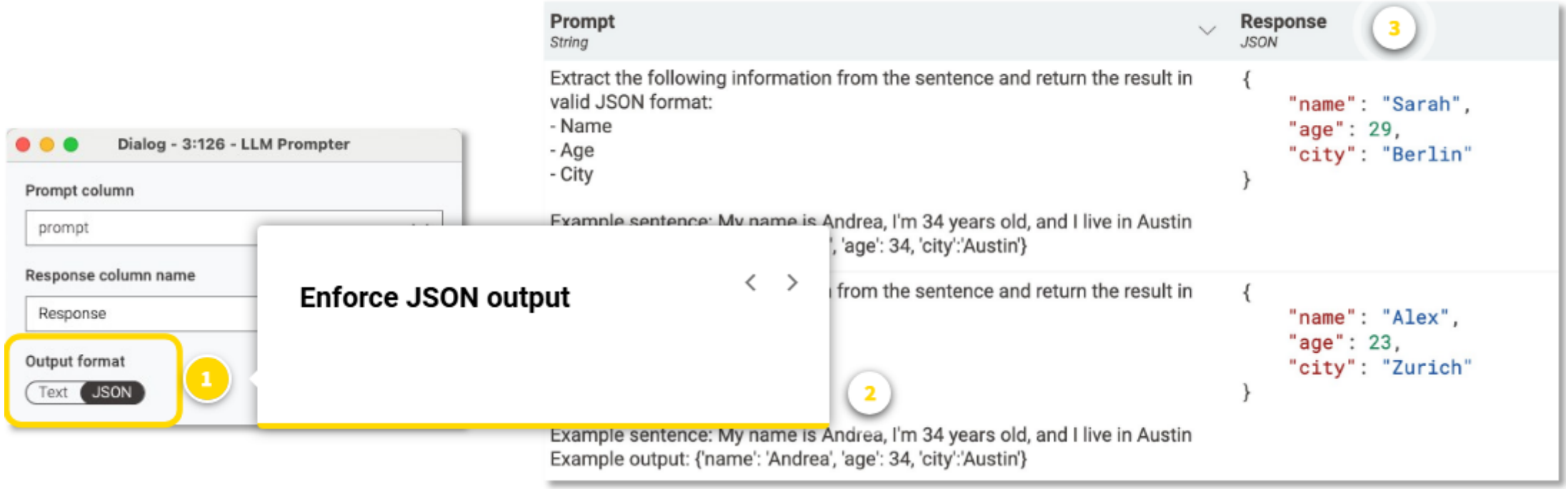
- So far, we've seen LLMs generating responses in plain text. However, LLMs can also generate structured output.
- The ability of LLMs to generate structured outputs such as valid JSON is essential for many downstream applications that need to reliably parse and use the model's responses.
- For example, if we want to analyze free-text customer support tickets in a dashboard by issue type, product, and urgency, the LLM must extract these fields in a structured format such as JSON to enable consistent parsing and downstream reporting and visualization.
- To enable the structured output:
  - Make sure you use the model that supports it (not all the models do),
  - Include the instruction to provide output in JSON in the prompt, if possible provide the examples,
  - Select the JSON mode in the output format configuration setting of the LLM Prompter node.

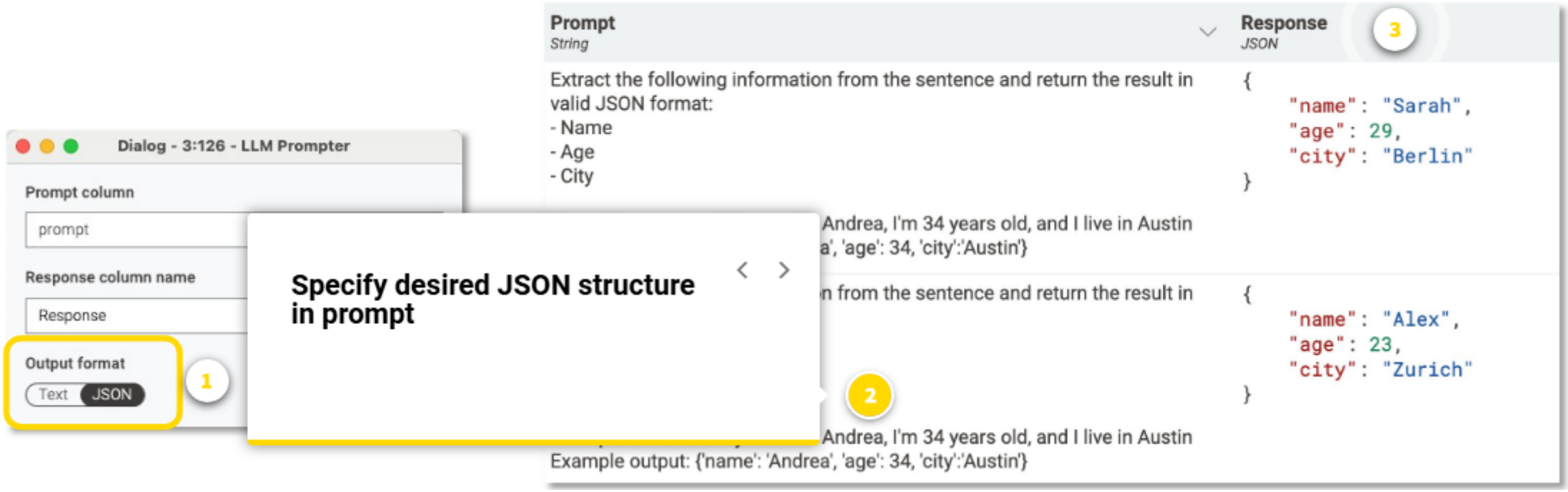


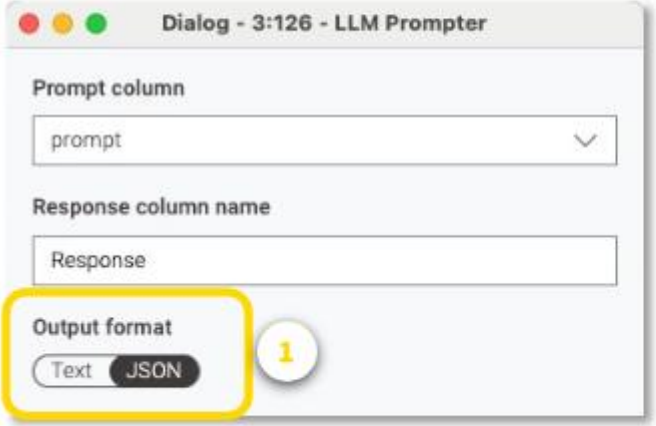
Prompt *String*	Response *JSON* 3

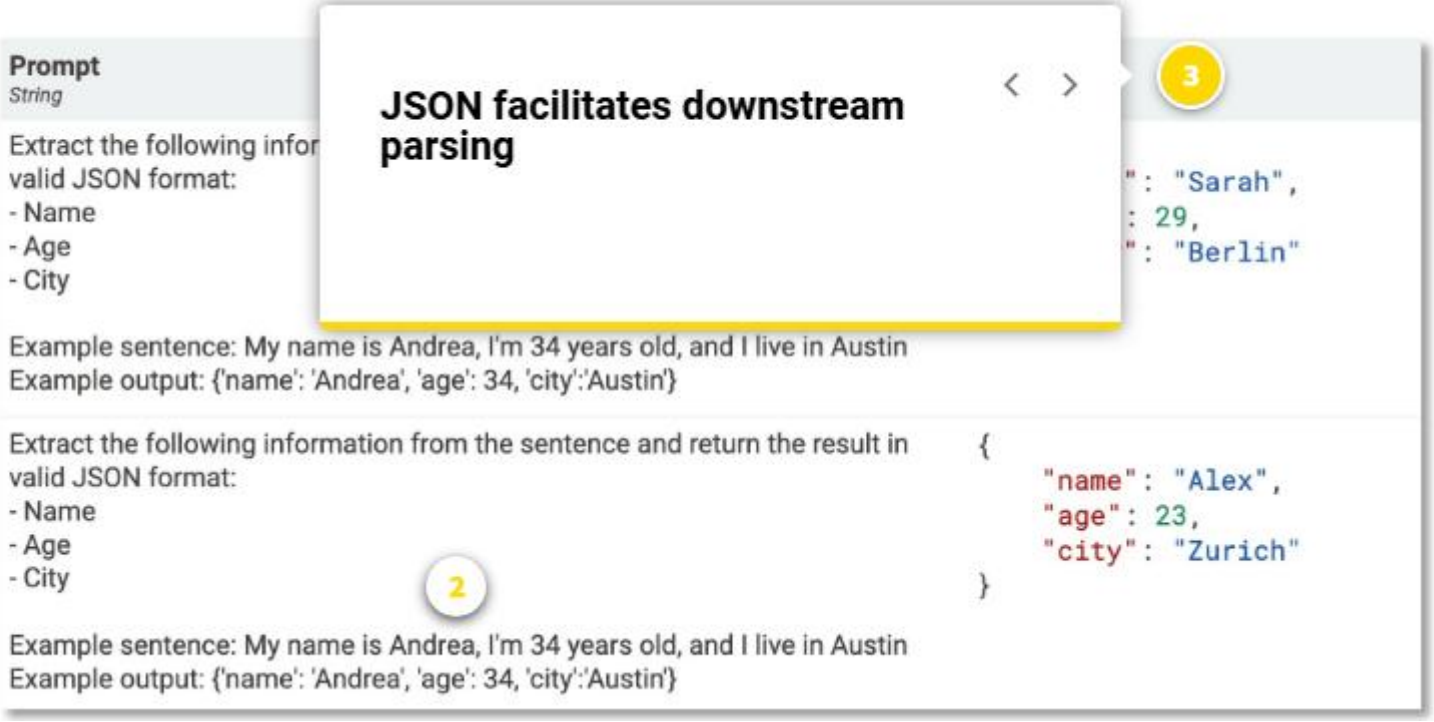
Extract the following information from the sentence and return the result in valid JSON format:
- Name
- Age
- City
 	
Example sentence: My name is Andrea, I'm 34 years old, and I live in Austin
Example output: {'name': 'Andrea', 'age': 34, 'city':'Austin'}
	{
	"name": "Sarah",
	"age": 29,
	"city": "Berlin"
	}

Extract the following information from the sentence and return the result in valid JSON format:
- Name
- Age
- City 2
 	
Example sentence: My name is Andrea, I'm 34 years old, and I live in Austin
Example output: {'name': 'Andrea', 'age': 34, 'city':'Austin'}
	{
	"name": "Alex",
	"age": 23,
	"city": "Zurich"
	}


Dialog - 3:126 - LLM Prompter
Prompt column
prompt
Response column name
Response
Output format
Text
JSON
1
Enforce JSON output
Prompt
String
Response
JSON
3
Extract the following information from the sentence and return the result in valid JSON format:
- Name
- Age
- City
Example sentence: My name is Andrea, I'm 34 years old, and I live in Austin
, 'age': 34, 'city':'Austin'}
{
"name": "Sarah",
"age": 29,
"city": "Berlin"
}
from the sentence and return the result in
2
Example sentence: My name is Andrea, I'm 34 years old, and I live in Austin
Example output: {'name': 'Andrea', 'age': 34, 'city':'Austin'}
{
"name": "Alex",
"age": 23,
"city": "Zurich"
}


Dialog - 3:126 - LLM Prompter
Prompt column
prompt
Response column name
Response
Output format
Text
JSON
1
Specify desired JSON structure in prompt
2
Prompt
String
Response
JSON
3
Extract the following information from the sentence and return the result in valid JSON format:
- Name
- Age
- City
Andrea, I'm 34 years old, and I live in Austin
a', 'age': 34, 'city':'Austin'}
n from the sentence and return the result in
Andrea, I'm 34 years old, and I live in Austin
Example output: {'name': 'Andrea', 'age': 34, 'city':'Austin'}
{
"name": "Sarah",
"age": 29,
"city": "Berlin"
}
{
"name": "Alex",
"age": 23,
"city": "Zurich"
}



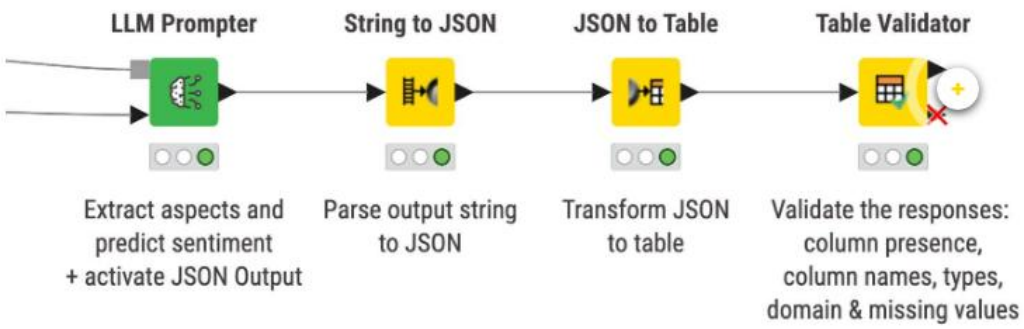


JSON facilitates downstream parsing

Output validation

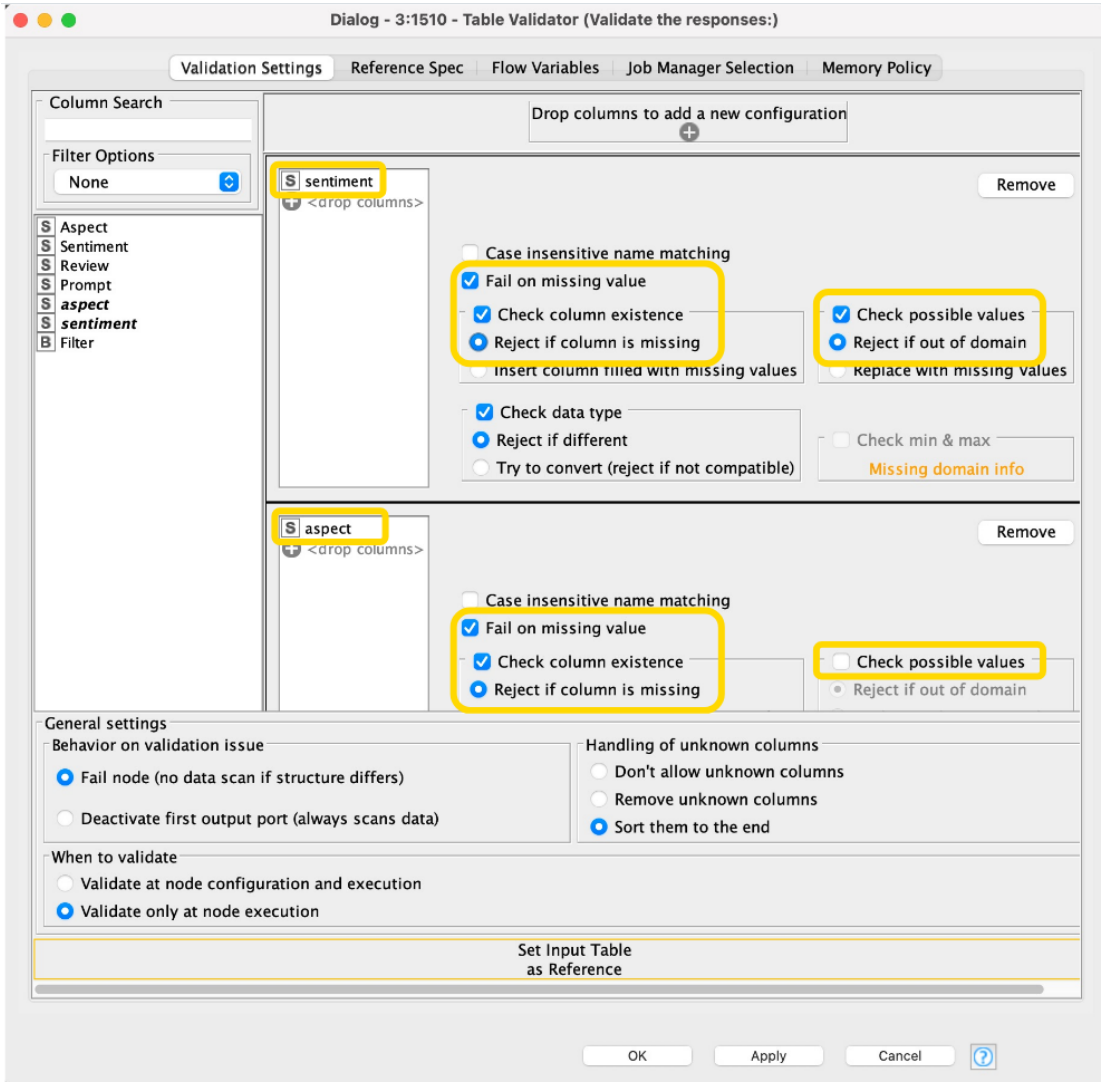
- Since LLMs aren’t deterministic in their responses, it is important to validate whether the response matches your expectations, especially in case we plan to use the structured output for downstream applications that need to reliably parse and use the model’s responses.
- In KNIME, we can validate structured output using native nodes such as the Table Validator.
- It helps perform a sanity check on the structure and content of the output.
- With the String Cleaner and String Format Manager node we can correct formatting issues and clean up text.

Output validation



- In cases where the model's response is incorrect or incomplete, you can implement error handling, for example:
- You can loop the response back to the model, asking it to correct itself. This technique allows you to refine the results interactively until they meet the desired criteria. You can also handle missing values during this process by checking for them and prompting the model to regenerate any incomplete content.
- To manage data types and formats, KNIME provides tools to parse output into the correct structure, whether converting strings into numbers, JSON, or other formats.
- Finally, you can set up breakpoints or conditions to determine when the output is satisfactory and the process can move forward.
- This kind of structured validation and error handling ensures high-quality, reliable results when integrating LLMs into your workflows.

Expanding Table Validator



Output validation

In cases where the model's response is incorrect or incomplete, we can implement error handling, for example:

- We can loop the response back to the model, asking it to correct itself.
- This technique allows you to refine the results interactively until they meet the desired criteria.
- We can also handle missing values during this process by checking for them and prompting the model to regenerate any incomplete content.
- To manage data types and formats, KNIME provides tools to parse output into the correct structure, whether converting strings into numbers, JSON, or other formats.
- Finally, we can set up breakpoints or conditions to determine when the output is satisfactory and the process can move forward.
- This kind of structured validation and error handling ensures high-quality, reliable results when integrating LLMs into your workflows.