Install Required Packages

```
! pip install -q --upgrade google-generativeai langchain-google-genai python-dotenv
```

```
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 50.4/50.4 kB 1.6 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 399.9/399.9 kB 14.6 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 290.2/290.2 kB 21.4 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 76.4/76.4 kB 5.1 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 77.9/77.9 kB 4.0 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 141.9/141.9 kB 11.2 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.3/58.3 kB 4.5 MB/s eta 0:00:00
```

```
#///// prompt: create a .env file in the workspace
# https://makersuite.google.com/

!echo -e 'GOOGLE_API_KEY=AIzaSyCkC-095CLMKV-wxSe2JDAtZJgTaJDskek' > .env
```

```
!ls -a
```

```
.  ..  .config  .env  sample_data
```

```
from dotenv import load_dotenv
load_dotenv()
```

```
True
```

```
from IPython.display import display
from IPython.display import Markdown
import textwrap


def to_markdown(text):
  text = text.replace('•', '  *')
  return Markdown(textwrap.indent(text, '> ', predicate=lambda _: True))
```

```
import google.generativeai as genai
import os
```

```
import os
genai.configure(api_key=os.environ.get("GOOGLE_API_KEY"))
```

Text Generation

```
model = genai.GenerativeModel(model_name = "gemini-pro")
model
```

```
genai.GenerativeModel(
    model_name='models/gemini-pro',
    generation_config={},
    safety_settings={},
    tools=None,
    system_instruction=None,
    cached_content=None
)
```

```
prompt = [
    "What is Mixture of Experts?",
]

response = model.generate_content(prompt)
```

```
to_markdown(response.text)
```

Mixture of Experts (MoE) is a machine learning model ensemble method that combines the predictions of multiple sub-models, known as "experts," to make a final prediction.

Key Concepts:

- **Experts:** Individual sub-models that specialize in different aspects of the input data.
- **Gating Network:** A model that determines which experts are responsible for making predictions for a given input.
- **Exponential Cascade:** A hierarchical structure where the predictions of one expert are passed as input to another expert.

How it Works:

1. **Input:** The input data is presented to the gating network.
2. **Gating Network:** The gating network assigns a weight to each expert, indicating its importance for the given input.
3. **Experts:** The input data is distributed to each expert, weighted by the gating network.
4. **Predictions:** Each expert makes a prediction based on its assigned input.
5. **Aggregation:** The predictions from all experts are combined using a weighted average or other aggregation method.
6. **Output:** The final prediction is the aggregated result.

Advantages:

- **Improved Performance:** By combining the predictions of multiple experts, MoE enhances the overall accuracy and generalization of the model.
- **Interpretability:** Assigning a weight to each expert provides insight into the importance and contribution of different sub-models in making predictions.
- **Parallelism:** The individual experts can be trained and executed in parallel, which improves training time and computational efficiency.

Applications:

MoE has been successfully applied in various domains, including:

- Natural Language Processing (NLP)
- Computer Vision
- Speech Recognition
- Recommender Systems
- Anomaly Detection

```
# response.prompt_feedback
# response.candidates
```

Use LangChain to Access Gemini API

```
from langchain_google_genai import ChatGoogleGenerativeAI
```

```
llm = ChatGoogleGenerativeAI(model="gemini-pro", google_api_key="AIzaSyDW_n54TMnahHUp9pMmKu0SjvhujCkSB7I")
```

```
result = llm.invoke("What is the sky blue ?")
```

```
to_markdown(result.content)
```

The sky is blue because of a phenomenon called Rayleigh scattering.

Sunlight is made up of all the colors of the visible spectrum. When sunlight passes through the atmosphere, the different colors of light are scattered in all directions by molecules of nitrogen and oxygen. The amount of scattering depends on the wavelength of the light, with shorter wavelengths (blue light) being scattered more than longer wavelengths (red light).

This is because the molecules in the atmosphere are much smaller than the wavelength of visible light. When light hits a molecule, it causes the electrons in the molecule to vibrate. The vibrating electrons then emit light in all directions, but the amount of light emitted depends on the wavelength of the light. Shorter wavelengths (blue light) are emitted more strongly than longer wavelengths (red light).

As a result of Rayleigh scattering, the sky appears blue during the day. At sunset and sunrise, the sunlight has to travel through

Chat

```
model = genai.GenerativeModel("gemini-1.5-flash")
chat = model.start_chat(history=[])
```

```
chat
```

```
ChatSession(
    model=genai.GenerativeModel(
        model_name='models/gemini-1.5-flash',
        generation_config={},
        safety_settings={},
        tools=None,
        system_instruction=None,
        cached_content=None
    ),
    history=[]
)
```

```
response = chat.send_message(
    "In one sentence, explain how a computer works to a young child."
)
to_markdown(response.text)
```

A computer is like a really smart toy that follows instructions and shows you pictures and videos, but it can do way more than

```
chat.history
```

```
[parts {
   text: "In one sentence, explain how a computer works to a young child."
 }
 role: "user",
 parts {
   text: "A computer is like a really smart toy that follows instructions and shows you pictures and videos, but it can do way more than that! \n"
 }
 role: "model"]
```

```
response = chat.send_message(
    "Okay, how about a more detailed explanation to a high schooler?", stream=True
)

for chunk in response:
    print(chunk.text)
    print("_" * 80)
```

```
A computer
________________________________________________________________________________
 fundamentally works by processing information, like a super-fast calculator, using a language
________________________________________________________________________________
 of ones and zeros called binary code. This code instructs the computer to perform operations on data,
________________________________________________________________________________
 like displaying images, running programs, or connecting to the internet, ultimately allowing us to interact with

________________________________________________________________________________
```

```
for message in chat.history:
    display(to_markdown(f"**{message.role}**: {message.parts[0].text}"))
```

user: In one sentence, explain how a computer works to a young child.

model: A computer is like a really smart toy that follows instructions and shows you pictures and videos, but it can do way more than that!

user: Okay, how about a more detailed explanation to a high schooler?

model: A computer fundamentally works by processing information, like a super-fast calculator, using a language of ones and zeros called binary code. This code instructs the computer to perform operations on data, like displaying images, running

Image & Text Generation

```
!curl -o image.jpg https://t0.gstatic.com/licensed-image?q=tbn:ANd9GcQ_Kevbk21QBRy-PgB4kQpS79brbmmEG7m3VOTShAn4PecDU5H5l
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  405k  100  405k    0     0  5866k      0 --:--:-- --:--:-- --:--:-- 5956k
```

```
import PIL.Image

img = PIL.Image.open("image.jpeg")
img
```

```
model = genai.GenerativeModel("gemini-1.5-flash")
```

```
response = model.generate_content(img)

to_markdown(response.text)
```

This image is of a character from the popular Japanese anime series Dragon Ball Z. The character is Goku, and he is depicted in a powerful pose with his hair flowing and a determined look on his face. The image is full of vibrant colors and dynamic energy, reflecting the intense battles and power levels found in the series.

```
response = model.generate_content(
    [
        "Write a short, engaging blog post based on this picture. It should include a description of the meal in the p
        img,
    ],
    stream=True,
)
response.resolve()
```

```
to_markdown(response.text)
```

I can't write a blog post about a picture of Goku, because I don't have access to any images. Please provide a picture of the meal so I can write a blog post about your meal prepping journey.

Chat with Documents

↳ *1 cell hidden*