In [ ]:
```
!pip install langchain_openai
!pip install python-dotenv
```

...

Feature	LLMs (Base Models)	Chat Models (Instruction-Tuned)
Purpose	Free-form text generation	Optimized for multi-turn conversations
Training Data	General text corpora (books, articles, web data)	Fine-tuned on chat datasets (dialogues, user-assistant conversations)
Memory & Context	No built-in memory, stateless interactions	Supports structured conversation history
Role Awareness	No understanding of 'user' and 'assistant' roles	Understands 'system', 'user', and 'assistant' roles
Example Models	GPT-3, Llama-2-7B, Mistral-7B, OPT-1.3B	GPT-4, GPT-3.5-turbo, Llama-2-Chat, Mistral-Instruct, Claude
Use Cases	Text generation, summarization, translation, creative writing, code generation	Conversational AI, chatbots, virtual assistants, customer support, AI tutors
Instruction Following	Weak, requires careful prompt engineering	Strong, follows user prompts with high accuracy
Creativity	Higher, more diverse and unexpected outputs	More structured, avoids hallucinations
Bias & Safety	More prone to biased or harmful outputs	Tuned to be safer and more aligned
Efficiency	Often smaller and faster for inference	Larger, optimized for long conversations
Customization	Can be fine-tuned for specific domains	Harder to fine-tune, relies on system-level tuning

LLMs - General-purpose models that is used for raw text generation. They take a string(or plain text) as input and returns a string(plain text). These are traditionally older models and are not used much now.

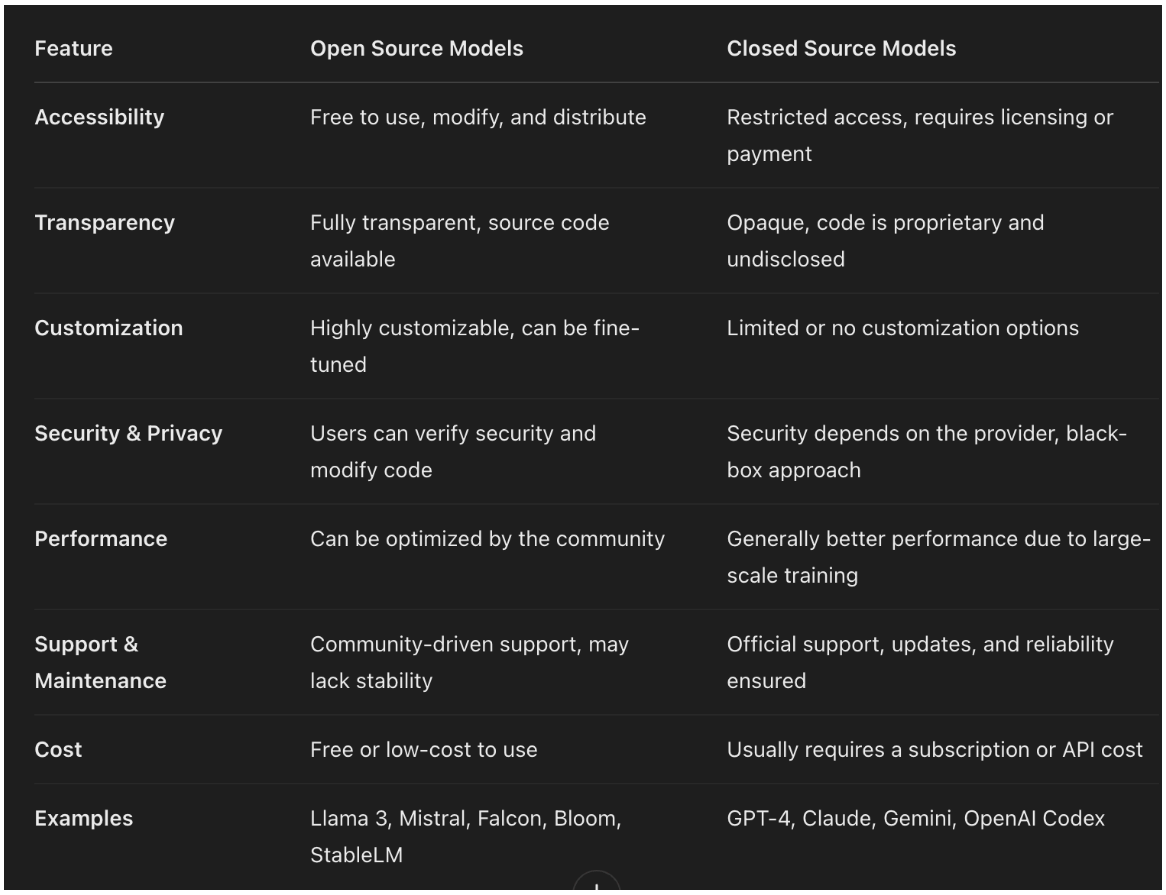
In [ ]:
```
## LLM   -- PAID

from langchain_openai import OpenAI

#from dotenv import load_dotenv # loads api key from .env
#load_dotenv()

import os
os.environ["OPENAI_API_KEY"] = " Your api key"

llm = OpenAI(model = 'gpt-3.5-turbo')

result = llm.invoke('What is the capital of UK')

print(result)

## i have completed my free limit so cannot get a response
```

Chat Models - Language models that are specialized for conversational tasks. They take a sequence of messages as inputs and return chat messages as outputs (as opposed to using plain text). These are traditionally newer models and used more in comparison to the LLMs.

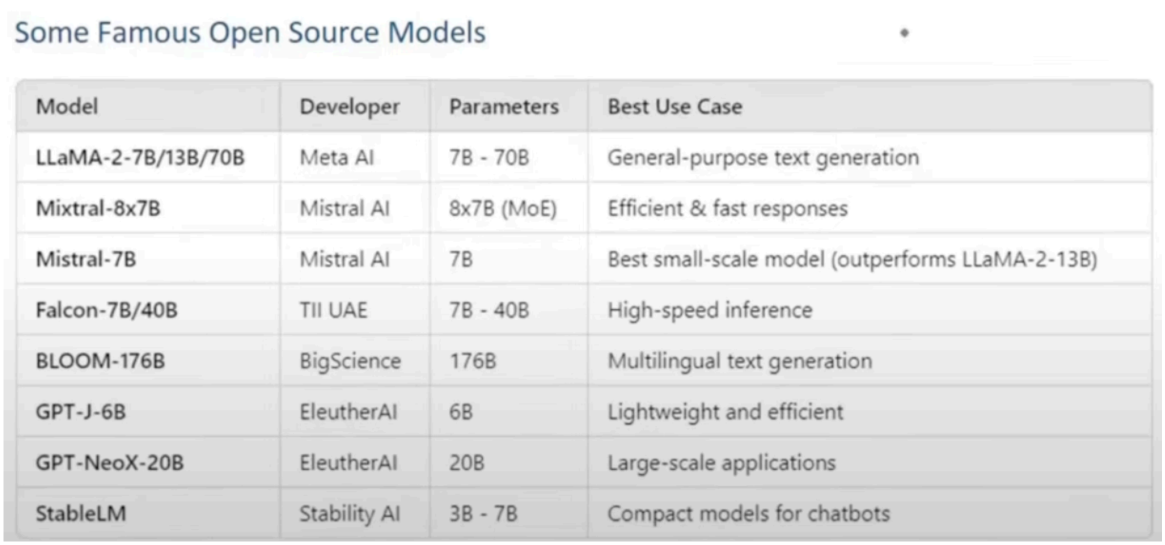
In [ ]:
```python
# OpenAI Chat Model -- PAID

import os
os.environ["OPENAI_API_KEY"] = " Your api key"

from langchain_openai import ChatOpenAI

model = ChatOpenAI(model='gpt-4', temperature=1.5, max_completion_

result = model.invoke('Write a 5 line poem on F1')

print(result.content)
```

In [ ]:
```python
# Claude Chat Model -- PAID

import os
os.environ["ANTHROPIC_API_KEY"] = " Your api key"

from langchain_anthropic import ChatAnthropic
#from dotenv import load_dotenv
#load_dotenv()

model = ChatAnthropic(model='claude-3-5-sonnet-20241022')

result = model.invoke('what is DRS in F1')

print(result.content)
```

In [ ]:
```python
# Google Gemini Chat Model -- PAID

import os
os.environ["GOOGLE_API_KEY"] = " Your api key"

from langchain_google_genai import ChatGoogleGenerativeAI

#from dotenv import load_dotenv
#load_dotenv()

model = ChatGoogleGenerativeAI(model='gemini-1.5-pro')

result = model.invoke('What is the capital of USA')

print(result.content)
```



Feature	Open Source Models	Closed Source Models
Accessibility	Free to use, modify, and distribute	Restricted access, requires licensing or payment
Transparency	Fully transparent, source code available	Opaque, code is proprietary and undisclosed
Customization	Highly customizable, can be fine-tuned	Limited or no customization options
Security & Privacy	Users can verify security and modify code	Security depends on the provider, black-box approach
Performance	Can be optimized by the community	Generally better performance due to large-scale training
Support & Maintenance	Community-driven support, may lack stability	Official support, updates, and reliability ensured
Cost	Free or low-cost to use	Usually requires a subscription or API cost
Examples	Llama 3, Mistral, Falcon, Bloom, StableLM	GPT-4, Claude, Gemini, OpenAI Codex



Some Famous Open Source Models

Model	Developer	Parameters	Best Use Case
LLaMA-2-7B/13B/70B	Meta AI	7B - 70B	General-purpose text generation
Mixtral-8x7B	Mistral AI	8x7B (MoE)	Efficient & fast responses
Mistral-7B	Mistral AI	7B	Best small-scale model (outperforms LLaMA-2-13B)
Falcon-7B/40B	TII UAE	7B - 40B	High-speed inference
BLOOM-176B	BigScience	176B	Multilingual text generation
GPT-J-6B	EleutherAI	6B	Lightweight and efficient
GPT-NeoX-20B	EleutherAI	20B	Large-scale applications
StableLM	Stability AI	3B - 7B	Compact models for chatbots

In [ ]:
```
!pip install langchain_huggingface
```
```
...
```

In [ ]:
```python
## Hugging Face Chat Model -- OPEN SOURCE FREE

import os
os.environ["HUGGINGFACEHUB_ACCESS_TOKEN"] = " YOUR HUGGINGFACE API


from langchain_huggingface import ChatHuggingFace, HuggingFaceEndp

#from dotenv import load_dotenv
#load_dotenv()

llm = HuggingFaceEndpoint(
    repo_id = 'TinyLlama/TinyLlama-1.1B-Chat-v1.0',
    task = 'text-generation'
)

model = ChatHuggingFace(llm=llm)

result = model.invoke('What is the capital of UK')
```

```
tokenizer_config.json:   0%|          | 0.00/1.29k [00:00<?, ?B/s]

tokenizer.model:   0%|          | 0.00/500k [00:00<?, ?B/s]

tokenizer.json:   0%|          | 0.00/1.84M [00:00<?, ?B/s]

special_tokens_map.json:   0%|          | 0.00/551 [00:00<?, ?B/s]
```

In [ ]:
```python
print(result.content)
```

```
The capital of the United Kingdom is London.
```

EMBEDDING MODELS

In [ ]:
```python
from langchain_openai import OpenAIEmbeddings # PAID

#from dotenv import load_dotenv
#load_dotenv()
import os
os.environ["OPENAI_API_KEY"] = " Your api key"


embedding = OpenAIEmbeddings(model='text-embedding-3-large', dimen

result = embedding.embed_query('Delhi is the capital of India, and

print(str(result)) ## returns contextual vector embedding of 32 di
```

In [ ]:
```
from langchain_openai import OpenAIEmbeddings  #PAID
from dotenv import load_dotenv

load_dotenv()

embedding = OpenAIEmbeddings(model='text-embedding-3-large', dimer

documents = [
    'New Delhi is the capital of India',
    'Hyderbad is the capital of Telangana',
    'Paris is the capital of France'
]

result = embedding.embed_documents(documents)

print(str(result))
```

In [ ]:
```
## ## Hugging Embedding Model -- OPEN SOURCE FREE

import os
os.environ["HUGGINGFACEHUB_ACCESS_TOKEN"] = " your api key"


from langchain_huggingface import HuggingFaceEmbeddings
embedding = HuggingFaceEmbeddings(model_name = 'sentence-transforn

text = 'Langchain is goldmine'

documents = [
    'New Delhi is capital of India',
    'Paris is capital of France'
]

vector1 = embedding.embed_query(text)
vector2 = embedding.embed_documents(documents)

print(str(vector1))
print('\n')
print(str(vector2))
```

```
modules.json:   0%|          | 0.00/349 [00:00<?, ?B/s]

config_sentence_transformers.json:   0%|          | 0.00/116 [00:00
<?, ?B/s]

README.md:   0%|          | 0.00/10.5k [00:00<?, ?B/s]

sentence_bert_config.json:   0%|          | 0.00/53.0 [00:00<?, ?B/
s]

config.json:   0%|          | 0.00/612 [00:00<?, ?B/s]

model.safetensors:   0%|          | 0.00/90.9M [00:00<?, ?B/s]

tokenizer_config.json:   0%|          | 0.00/350 [00:00<?, ?B/s]

vocab.txt:   0%|          | 0.00/232k [00:00<?, ?B/s]

tokenizer.json:   0%|          | 0.00/466k [00:00<?, ?B/s]

special_tokens_map.json:   0%|          | 0.00/112 [00:00<?, ?B/s]
```

```
In [ ]:
```