Using AI Foundry Models for AGAthon 2025

1 Getting Started with Azure AI Foundry

Welcome to the AGAthon 2025! This guide will help you use the AI Foundry models available for your healthcare AI solutions.

---

2 Prerequisites and Installation

Required Python Packages

Create a requirements.txt file with the following dependencies:

```
ipykernel
langchain
langchain-openai
python-dotenv
scikit-learn
azure-ai-projects
azure-identity
openai
```

Install all packages:

```
pip install -r requirements.txt
```

---

3 Environment Configuration

3.1 Setting Up Your .env File

Create a .env file in your project root directory with the following environment variables:

```
OPENAI_API_TYPE=azure
OPENAI_API_VERSION=[API_VERSION]
OPENAI_API_KEY=[YOUR_API_KEY]
AZURE_OPENAI_API_KEY=[YOUR_AZURE_API_KEY]
AZURE_OPENAI_ENDPOINT=[YOUR_AZURE_ENDPOINT]
```

3.2 Loading Environment Variables

At the beginning of your Python scripts, always load your environment variables:

```python
from dotenv import load_dotenv
import os

# Load environment variables from .env file
load_dotenv()

# Access variables
api_key = os.getenv("AZURE_OPENAI_API_KEY")
endpoint = os.getenv("AZURE_OPENAI_ENDPOINT")
api_version = os.getenv("OPENAI_API_VERSION")
```

4 Common Model Usage Examples

4.1 GPT-4o-mini - General Purpose Chat Model

Best for: Quick responses, general medical queries, data summarization

```
import os
from openai import AzureOpenAI

endpoint = "https://agathonaifoundry.cognitiveservices.azure.com/"
model_name = "gpt-4o-mini"
deployment = "gpt-4o-mini"

subscription_key = "<your-api-key>"
api_version = "2024-12-01-preview"

client = AzureOpenAI(
    api_version=api_version,
    azure_endpoint=endpoint,
    api_key=subscription_key,
)

response = client.chat.completions.create(
    messages=[
        {
            "role": "system",
            "content": "You are a helpful assistant.",
        },
        {
            "role": "user",
            "content": "I am going to Paris, what should I see?",
        }
    ],
    max_tokens=4096,
    temperature=1.0,
    top_p=1.0,
    model=deployment
)

print(response.choices[0].message.content)
```

URL	https://agathonaifoundry.cognitiveservices.azure.com/openai/deployments/gpt-4o-mini/chat/completions?api-version=2025-01-01-preview
Key	ASqWh80z8lo1RGN0ogGH7PdQtukNdv4DBdh9LslJcFob8ezs788DJQQJ99BJACPV0roXJ3w3AAAAACOG-DHW9
Model Name	gpt-4o-mini
API Version	2025-01-01-preview

4.2 GPT-4.1 - Advanced Reasoning and Analysis

Best for: Complex medical analysis, differential diagnosis, treatment planning

```
import os
from openai import AzureOpenAI

endpoint = "https://agathonaifoundry.cognitiveservices.azure.com/"
model_name = "gpt-4.1"
deployment = "gpt-4.1"

subscription_key = "<your-api-key>"
api_version = "2024-12-01-preview"

client = AzureOpenAI(
    api_version=api_version,
    azure_endpoint=endpoint,
    api_key=subscription_key,
)

response = client.chat.completions.create(
    messages=[
        {
            "role": "system",
            "content": "You are a helpful assistant.",
        },
        {
            "role": "user",
            "content": "I am going to Paris, what should I see?",
        }
    ],
    max_completion_tokens=13107,
    temperature=1.0,
    top_p=1.0,
    frequency_penalty=0.0,
    presence_penalty=0.0,
    model=deployment
)

print(response.choices[0].message.content)
```

URL	https://agathonaifoundry.cognitiveservices.azure.com/openai/deployments/gpt-4.1/chat/comple-tions?api-version=2025-01-01-preview
Key	ASqWh80z8lo1RGN0ogGH7PdQtukNdv4DBdh9LslJcFob8ezs788DJQQJ99BJACPV0roXJ3w3AAAAACOG-DHW9
Model Name	gpt-4.1
API Ver-sion	2025-01-01-preview

4.3 GPT-3.5-Turbo - Fast and Cost-Effective

Best for: High-volume processing, simple classifications, data extraction

```
import os
from openai import AzureOpenAI

endpoint = "https://nikla-mgt2rh71-eastus2.cognitiveservices.az-
ure.com/"
model_name = "gpt-35-turbo"
deployment = "gpt-35-turbo"

subscription_key = "<your-api-key>"
api_version = "2024-12-01-preview"

client = AzureOpenAI(
    api_version=api_version,
    azure_endpoint=endpoint,
    api_key=subscription_key,
)

response = client.chat.completions.create(
    messages=[
        {
            "role": "system",
            "content": "You are a helpful assistant.",
        },
        {
            "role": "user",
            "content": "I am going to Paris, what should I see?",
        }
    ],
    max_tokens=4096,
    temperature=1.0,
    top_p=1.0,
    model=deployment
)

print(response.choices[0].message.content)
```

URL	https://nikla-mgt2rh71-eastus2.cognitiveservices.azure.com/openai/de-ployments/gpt-35-turbo/chat/completions?api-version=2025-01-01-preview
Key	2iJeCOpsykTlEMtS3mK8hDTENJS6tK274iZAG2Q7BiPl59IIpdekJQQJ99BJA-CHYHv6XJ3w3AAAAACOGwpVD
Model Name	gpt-35-turbo
API Version	2025-01-01-preview

4.4 text-embedding-3-large - Text Embeddings (For Semantic Search)

Best for: Document similarity, semantic search, clustering medical records

```
import os
from openai import AzureOpenAI

endpoint = os.getenv("AZURE_OPENAI_ENDPOINT"),
model_name = "text-embedding-3-large"

api_version = os.getenv("OPENAI_API_VERSION"),

client = AzureOpenAI(
    api_version=api_version
    endpoint=endpoint,
    credential= os.getenv("AZURE_OPENAI_API_KEY"),
)

response = client.embeddings.create(
    input=["first phrase","second phrase","third phrase"],
    model=model_name
)

for item in response.data:
    length = len(item.embedding)
    print(
        f"data[{item.index}]: length={length}, "
        f"[{item.embedding[0]}, {item.embedding[1]}, "
        f"..., {item.embedding[length-2]}, {item.embedding[length-
1]}]"
    )
print(response.usage)
```

4.5 GPT-4o-mini-transcribe

```
import os
from openai import AzureOpenAI

endpoint = os.getenv("AZURE_OPENAI_ENDPOINT", "https://nikla-
mgt2rh71-eastus2.cognitiveservices.azure.com/")
deployment = os.getenv("AZURE_OPENAI_DEPLOYMENT", "gpt-4o-mini-
transcribe")
api_key = os.getenv("AZURE_OPENAI_API_KEY")
api_version = os.getenv("OPENAI_API_VERSION", "2025-03-01-pre-
view")

client = AzureOpenAI(
    api_version=api_version,
    azure_endpoint=endpoint,
    api_key=api_key,
)

# Open and transcribe the audio file
audio_file_path = os.getenv("AUDIO_FILE_PATH", "path/to/file/au-
dio.mp3")

with open(audio_file_path, "rb") as audio_file:
    response = client.audio.transcriptions.create(
        model=deployment,
        file=audio_file
    )

print(response.text)
```

URL	https://nikla-mgt2rh71-eastus2.cognitiveservices.azure.com/openai/deployments/gpt-4o-mini-transcribe/audio/transcriptions?api-version=2025-03-01-preview
Key	2iJeCOpsykTlEMtS3mK8hDTENJS6tK274iZAG2Q7BiPl59IlpdekJQQJ99BJACHYHv6XJ3w3AAAAACOGwpVD
Model Name	gpt-4o-mini-transcribe
API Version	2025-03-01-preview

4.6 GPT-4o-mini-tts

```
import os
from openai import AzureOpenAI

# Load environment variables
endpoint = os.getenv("AZURE_OPENAI_ENDPOINT", "https://nikla-
mgt2rh71-eastus2.cognitiveservices.azure.com/")
deployment = os.getenv("AZURE_OPENAI_TTS_DEPLOYMENT", "gpt-4o-
mini-tts")
api_key = os.getenv("AZURE_OPENAI_API_KEY")
api_version = os.getenv("OPENAI_API_VERSION", "2025-03-01-pre-
view")

client = AzureOpenAI(
    api_version=api_version,
    azure_endpoint=endpoint,
    api_key=api_key,
)

# Generate speech from text
text_input = os.getenv("TTS_INPUT", "The quick brown fox jumped
over the lazy dog")
# use this in case the audio cuts off at the start and in the end.
# text_input = "[pause] " + text_input + " [pause]"
voice = os.getenv("TTS_VOICE", "alloy")
output_file = os.getenv("TTS_OUTPUT_FILE", "output_speech.mp3")

with client.audio.speech.with_streaming_response.create(
    model=deployment,
    voice=voice,
    input=text_input
) as response:
     response.stream_to_file(output_file)
# Save the audio file
response.stream_to_file(output_file)
print(f"Audio saved to {output_file}")
```

URL	https://nikla-mgt2rh71-eastus2.cognitiveservices.azure.com/openai/deploy-ments/gpt-4o-mini-tts/audio/speech?api-version=2025-03-01-preview
Key	2iJeCOpsykTlEMtS3mK8hDTENJS6tK274iZAG2Q7BiPl59IIpdekJQQJ99BJACH-YHv6XJ3w3AAAAACOGwpVD
Model Name	gpt-4o-mini-tts
API Version	2025-03-01-preview

5 Advanced Model Parameters

5.1 Temperature Control

```
python
# Conservative (more deterministic) - Good for medical analysis
llm_conservative = AzureChatOpenAI(
    deployment_name="gpt-4o-mini",
    temperature=0.0,  # Most deterministic
)

# Balanced - Good for general responses
llm_balanced = AzureChatOpenAI(
    deployment_name="gpt-4o-mini",
    temperature=0.7,  # Balanced creativity
)

# Creative - Good for brainstorming solutions
llm_creative = AzureChatOpenAI(
    deployment_name="gpt-4o-mini",
    temperature=1.5,  # More creative/random
)
```

5.2 Token Limits and Response Control

```
llm = AzureChatOpenAI(
    deployment_name="gpt-4o-mini",
    max_tokens=500,      # Maximum response length
    top_p=0.95,          # Nucleus sampling (alternative to tem-
perature)
    frequency_penalty=0.0,  # Reduce repetition (0.0-2.0)
    presence_penalty=0.0,   # Encourage new topics (0.0-2.0)
)
```

6 Troubleshooting

6.1 Common Issues

Import Errors:

```
pip install --upgrade langchain langchain-openai
```

Authentication Errors:

```
# Verify environment variables are loaded
print(os.getenv("AZURE_OPENAI_ENDPOINT"))
print(os.getenv("AZURE_OPENAI_API_KEY")[:10] + "...")
```

Rate Limiting:

- Implement retry logic
- Reduce concurrent requests

7 Support and Resources

- **Technical Support**:
    - Niklas.Grimm@campana-schott.com
    - Marcel.Heidebrecht@campana-schott.com
    - Xuan-Xuyen.Nguyen@campana-schott.com
- **Model Documentation**: [DOCS_URL]
- **Code Examples Repository**: [GITHUB_URL]

Good luck with your AGAthon project!