Satellite Image Cloud Detection via AzureOpenAI Inference

1. Objective

To build a lightweight, user-friendly interface where users can upload a satellite image and instantly receive a label — "Cloudy" or "Clear" — by leveraging a Large Language Model (LLM) deployed on Azure OpenAI. The LLM is prompted to perform visual scene analysis and classification, removing the need for users to set up or maintain conventional deep learning models. This approach enables fast, API-driven cloud detection and simplifies workflows for users without machine learning or programming expertise.

2. Problem Statement

Manual inspection of satellite imagery to identify cloud-covered scenes is time-consuming and does not scale for large datasets. Traditional solutions require deploying or fine-tuning pretrained image classification models, which can be technically challenging for many users. There is a need for a simple web-based application that allows users to upload satellite images and instantly obtain a "Cloudy" or "Clear" classification. By leveraging the image understanding capabilities of modern LLMs via Azure OpenAI, we can bypass complex model management and provide an accessible, code-free cloud detection solution suitable for integration into various industries such as media, agriculture, logistics, and climate analytics.

Your task is to build a minimal application — CLI, script, or web-based (Streamlit) — that:

- Accepts satellite images as input (e.g., .jpg, .png)
- Uses LLM for inference
- Returns an output label: **"Cloudy"** or **"Clear"**
- Optionally, shows the image back with its predicted label

This inference-based approach saves time and enables **instant integration into operational pipelines** for media, agriculture, logistics, and climate analytics teams.

3. Inputs / Shared Artifacts

- Azure OpenAI Resource:
  - API key
  - Endpoint URL
  - Deployment name (to be set as environment variables)
- Sample images:
  - https://www.kaggle.com/datasets/hmendonca/cloud-cover-detection/data

4. Expected Outcome

For each uploaded satellite image, the system should:

- Return a **single label**: either **"Cloudy"** or **"Clear"**
- Optionally, display the **confidence score** (e.g., 92.4% confidence in prediction)
- Visually present the image back to the user with the predicted label shown

Example:

- **Input**: https://....jpg
- **Output**:
  - **Prediction**: Cloudy

5. Concepts Covered

- Azure OpenAI API usage
- Image Classification
- Open-source model
- Pass multimodal data to models

6. Example: Implementation

```
# pip install langchain-openai pillow requests
import base64
import io
import requests
from PIL import Image
from langchain_openai import AzureChatOpenAI
from pydantic import BaseModel, Field

# --- Azure OpenAI Config ---
os.environ["AZURE_OPENAI_ENDPOINT"] = "https://aiportalapi.stu-
platform.live/jpe"
os.environ["AZURE_OPENAI_API_KEY"] = ""
os.environ["AZURE_DEPLOYMENT_NAME"] = "GPT-4o-mini"
# --- Setup LLM ---
llm = AzureChatOpenAI(
    azure_endpoint=os.environ["AZURE_OPENAI_ENDPOINT"],
    azure_deployment=os.environ["AZURE_DEPLOYMENT_NAME"],
    api_key=os.environ["AZURE_OPENAI_API_KEY"],
    api_version="2024-02-15-preview",
)

# --- Output Schema ---
class WeatherResponse(BaseModel):
    accuracy: float = Field(description="The accuracy of the result")
    result: str = Field(description="The result of the classification")

llm_with_structured_output = llm.with_structured_output(WeatherResponse)

# --- Load image from URL ---
image_url = "https://images.pexels.com/photos/53594/blue-clouds-day-fluffy-
53594.jpeg?cs=srgb&dl=pexels-pixabay-53594.jpg&fm=jpg"
response = requests.get(image_url)
image_bytes = response.content
image_data_base64 = base64.b64encode(image_bytes).decode("utf-8")

# --- Prompt Construction ---
message = [
    {
        "role": "system",
        "content": """Based on the satellite image provided, classify the
scene as either:
        'Clear' (no clouds) or 'Cloudy' (with clouds).
        Respond with only one word: either 'Clear' or 'Cloudy' and Accuracy.
Do not provide explanations.""",
    },
    {
        "role": "user",
        "content": [
            {
                "type": "text",
```

```
                "text": "Classify the scene as either: 'Clear' or 'Cloudy'
and Accuracy.",
            },
            {
                "type": "image_url",
                "image_url": {"url":
f"{image_data_base64}"},
            },
        ],
    },
]

# --- Call Azure OpenAI ---
try:
    result = llm_with_structured_output.invoke(message)
    print(f"Prediction: {result.result}")
    print(f"Accuracy:: {result.accuracy} %")
except Exception as e:
    print(f"Error: {e}")
```

7. Final Submission

- Complete source code:
    - Input with 3 mock images (via URLs). Do not use manual input.
    - Print logs in console
- Use a dummy input list (auto input). Do not use manual input (input() built-in).
- README file description: step by step to solve the problem
    - LangChain document source to support to pass multimodal data (image) to LLM
    - Knowledge, experience gained after the exercise