# Chapter 4: Advanced AWS AI/ML Services

This chapter covers more specialized and powerful AI/ML services offered by AWS. These services build upon the core services we've already explored and provide advanced capabilities for specific use cases.

## Prerequisites

* An AWS account with appropriate permissions
* AWS CLI configured with your credentials
* Python 3.7 or later
* Required Python packages: boto3, pandas, numpy, matplotlib, scikit-learn

Install required packages:

bash

Copy

pip install boto3 pandas numpy matplotlib scikit-learn

## 1. Amazon SageMaker Advanced Features

### 1.1 SageMaker Autopilot

SageMaker Autopilot automatically trains and tunes the best machine learning models for classification or regression based on your data.

python

Copy

import boto3

import sagemaker

import pandas as pd

from sagemaker.session import Session

from sagemaker.autopilot.automl import AutoML

from sklearn.datasets import load\_diabetes

from sklearn.model\_selection import train\_test\_split

*# Load sample data*

diabetes = load\_diabetes()

X = pd.DataFrame(diabetes.data, columns=diabetes.feature\_names)

y = pd.DataFrame(diabetes.target, columns=['target'])

data = pd.concat([X, y], axis=1)

*# Split the data*

train, test = train\_test\_split(data, test\_size=0.2)

*# Save data to CSV files*

train.to\_csv('train.csv', index=False)

test.to\_csv('test.csv', index=False)

*# Set up SageMaker session*

sagemaker\_session = sagemaker.Session()

bucket = sagemaker\_session.default\_bucket()

prefix = 'sagemaker-autopilot-diabetes'

*# Upload data to S3*

train\_s3 = sagemaker\_session.upload\_data('train.csv', bucket=bucket, key\_prefix=prefix)

test\_s3 = sagemaker\_session.upload\_data('test.csv', bucket=bucket, key\_prefix=prefix)

*# Set up Autopilot job*

auto\_ml = AutoML(

role='your-sagemaker-role-arn', *# Replace with your SageMaker role ARN*

target\_attribute\_name='target',

sagemaker\_session=sagemaker\_session,

output\_path=f's3://{bucket}/{prefix}/output'

)

*# Run Autopilot job*

auto\_ml.fit(

inputs=train\_s3,

job\_name='diabetes-autopilot-job',

wait=True

)

*# Get best model*

best\_model = auto\_ml.best\_candidate()

print(f"Best model: {best\_model}")

*# Deploy model*

predictor = auto\_ml.deploy(

initial\_instance\_count=1,

instance\_type='ml.m5.large',

candidate=best\_model

)

*# Make predictions*

test\_data = test.drop('target', axis=1)

predictions = predictor.predict(test\_data.values)

print("Sample predictions:")

print(predictions[:5])

*# Clean up*

predictor.delete\_endpoint()

### 1.2 SageMaker Model Monitor

SageMaker Model Monitor continuously monitors the quality of machine learning models in production.

python

Copy

import boto3

import sagemaker

from sagemaker.model\_monitor import DataCaptureConfig

from sagemaker.model\_monitor import DefaultModelMonitor

from sagemaker.model\_monitor.dataset\_format import DatasetFormat

*# Assume you have a deployed model endpoint named 'your-model-endpoint'*

endpoint\_name = 'your-model-endpoint'

*# Set up data capture*

data\_capture\_config = DataCaptureConfig(

enable\_capture=True,

sampling\_percentage=100,

destination\_s3\_uri=f's3://{bucket}/{prefix}/data-capture'

)

*# Create a default model monitor*

my\_monitor = DefaultModelMonitor(

role='your-sagemaker-role-arn', *# Replace with your SageMaker role ARN*

instance\_count=1,

instance\_type='ml.m5.xlarge',

volume\_size\_in\_gb=20,

max\_runtime\_in\_seconds=3600,

)

*# Set up the baseline*

my\_monitor.suggest\_baseline(

baseline\_dataset='s3://path-to-your-baseline-dataset/baseline.csv', *# Replace with your baseline dataset*

dataset\_format=DatasetFormat.csv(header=True),

output\_s3\_uri=f's3://{bucket}/{prefix}/baseline-results',

wait=True

)

*# Create a monitoring schedule*

from sagemaker.model\_monitor import CronExpressionGenerator

my\_monitor.create\_monitoring\_schedule(

monitor\_schedule\_name='my-monitoring-schedule',

endpoint\_input=endpoint\_name,

output\_s3\_uri=f's3://{bucket}/{prefix}/monitoring-output',

statistics=my\_monitor.baseline\_statistics(),

constraints=my\_monitor.suggested\_constraints(),

schedule\_cron\_expression=CronExpressionGenerator.hourly(),

enable\_cloudwatch\_metrics=True,

)

print("Model monitoring schedule created.")

## 2. Amazon Augmented AI (A2I)

Amazon A2I provides built-in human review workflows for common machine learning use cases. Here's an example of setting up a human review workflow for image moderation:

python

Copy

import boto3

import json

a2i = boto3.client('sagemaker-a2i-runtime')

s3 = boto3.client('s3')

*# Assume you have created a human review workflow in the A2I console*

*# and have the ARN of the flow definition*

flow\_definition\_arn = 'your-flow-definition-arn' *# Replace with your flow definition ARN*

*# Sample image URL*

image\_url = "https://example.com/image-to-moderate.jpg"

*# Create a human loop*

response = a2i.start\_human\_loop(

HumanLoopName='image-moderation-loop-' + str(int(time.time())),

FlowDefinitionArn=flow\_definition\_arn,

HumanLoopInput={

'InputContent': json.dumps({

"initialValue": "EXPLICIT",

"imageUrl": image\_url

})

}

)

print(f"Human loop created: {response['HumanLoopArn']}")

*# In a real scenario, you would wait for the human review to complete*

*# and then process the results. Here's how you might check the status:*

*# Get human loop status*

human\_loop\_name = response['HumanLoopArn'].split('/')[-1]

status\_response = a2i.describe\_human\_loop(HumanLoopName=human\_loop\_name)

print(f"Human loop status: {status\_response['HumanLoopStatus']}")

*# When the loop is completed, you can get the results from S3*

*# The results location would be specified in your flow definition*

## 3. Amazon Textract

Amazon Textract is a service that automatically extracts text, handwriting, and data from scanned documents. Here's an example of using Textract to extract text and form data from a document:

python

Copy

import boto3

import time

textract = boto3.client('textract')

s3 = boto3.client('s3')

*# Assume you have a document in your S3 bucket*

bucket\_name = 'your-bucket-name' *# Replace with your bucket name*

document\_name = 'sample-form.pdf' *# Replace with your document name*

*# Start the Textract job*

response = textract.start\_document\_analysis(

DocumentLocation={

'S3Object': {

'Bucket': bucket\_name,

'Name': document\_name

}

},

FeatureTypes=['FORMS', 'TABLES']

)

job\_id = response['JobId']

*# Wait for the job to complete*

while True:

response = textract.get\_document\_analysis(JobId=job\_id)

status = response['JobStatus']

print(f"Job status: {status}")

if status in ['SUCCEEDED', 'FAILED']:

break

time.sleep(5)

*# Process the results*

if status == 'SUCCEEDED':

print("\nExtracted text:")

for item in response['Blocks']:

if item['BlockType'] == 'LINE':

print(item['Text'])

print("\nExtracted form data:")

for item in response['Blocks']:

if item['BlockType'] == 'KEY\_VALUE\_SET':

if 'KEY' in item['EntityTypes']:

key = item['Relationships'][0]['Ids']

value = item['Relationships'][1]['Ids']

key\_text = next(block['Text'] for block in response['Blocks'] if block['Id'] == key[0])

value\_text = next(block['Text'] for block in response['Blocks'] if block['Id'] == value[0])

print(f"{key\_text}: {value\_text}")

else:

print("Textract job failed.")

## 4. Amazon Kendra

Amazon Kendra is an intelligent search service powered by machine learning. Here's an example of setting up a Kendra index and performing a search:

python

Copy

import boto3

import time

kendra = boto3.client('kendra')

*# Create a Kendra index*

response = kendra.create\_index(

Name='sample-index',

Edition='DEVELOPER\_EDITION',

RoleArn='your-kendra-role-arn' *# Replace with your Kendra role ARN*

)

index\_id = response['Id']

*# Wait for the index to be created*

while True:

response = kendra.describe\_index(Id=index\_id)

status = response['Status']

print(f"Index status: {status}")

if status == 'ACTIVE':

break

time.sleep(60)

*# Add a data source (example: S3)*

response = kendra.create\_data\_source(

IndexId=index\_id,

Name='sample-data-source',

Type='S3',

DataSourceConfiguration={

'S3Configuration': {

'BucketName': 'your-bucket-name', *# Replace with your bucket name*

'InclusionPrefixes': ['documents/']

}

},

RoleArn='your-kendra-role-arn' *# Replace with your Kendra role ARN*

)

data\_source\_id = response['Id']

*# Sync the data source*

kendra.start\_data\_source\_sync\_job(

Id=data\_source\_id,

IndexId=index\_id

)

*# Perform a search*

response = kendra.query(

IndexId=index\_id,

QueryText='What is machine learning?'

)

print("\nSearch results:")

for result in response['ResultItems']:

print(f"Document Title: {result.get('DocumentTitle')}")

print(f"Document Excerpt: {result.get('DocumentExcerpt', {}).get('Text')}")

print("---")

*# Clean up (uncomment to delete the index when you're done)*

*# kendra.delete\_index(Id=index\_id)*

## Conclusion

These advanced AWS AI/ML services provide powerful capabilities for specific use cases:

1. SageMaker Autopilot for automated machine learning
2. SageMaker Model Monitor for continuous model quality monitoring
3. Amazon Augmented AI (A2I) for human review workflows
4. Amazon Textract for document text and data extraction
5. Amazon Kendra for intelligent search

These services can significantly accelerate your AI/ML projects by automating complex tasks and providing specialized functionality. As you work with these services, remember to:

* Manage your AWS resources carefully to control costs
* Ensure you have the necessary permissions and IAM roles set up
* Be aware of service limits and quotas
* Consider data privacy and security, especially when dealing with sensitive information

As you become more familiar with these advanced services, explore how they can be combined with core AI services and your existing workflows to create sophisticated, AI-powered applications.