UDACITY

SageMaker essentials

Lamia Zain
lamiahasan4@gmail.com
LinkedIn

July,19, 2024

Connect Sessions | Purpose

A Connect Session **IS**:

- Focused on learning, encouragement & graduation for a group of students coached by a Udacity Session Lead
- Setting weekly study goals
- Helping each other with progress (including peer to peer)
- Keeping everyone accountable for their responsibilities
- A way to meet individuals in tech field & learn about the industry
- **Mandatory**

A Connect Session **IS NOT**:

- A social meetup
- A study group
- A substitute for online learning
- **Optional**

UDACITY

Let’s check your progress

You are encouraged to spend at lest 10 hours/week to graduate.

Presentation date

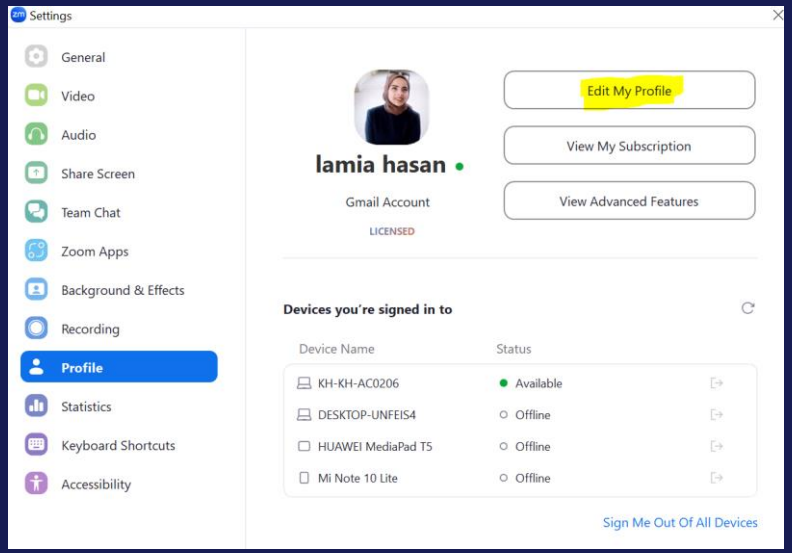
UDACITY

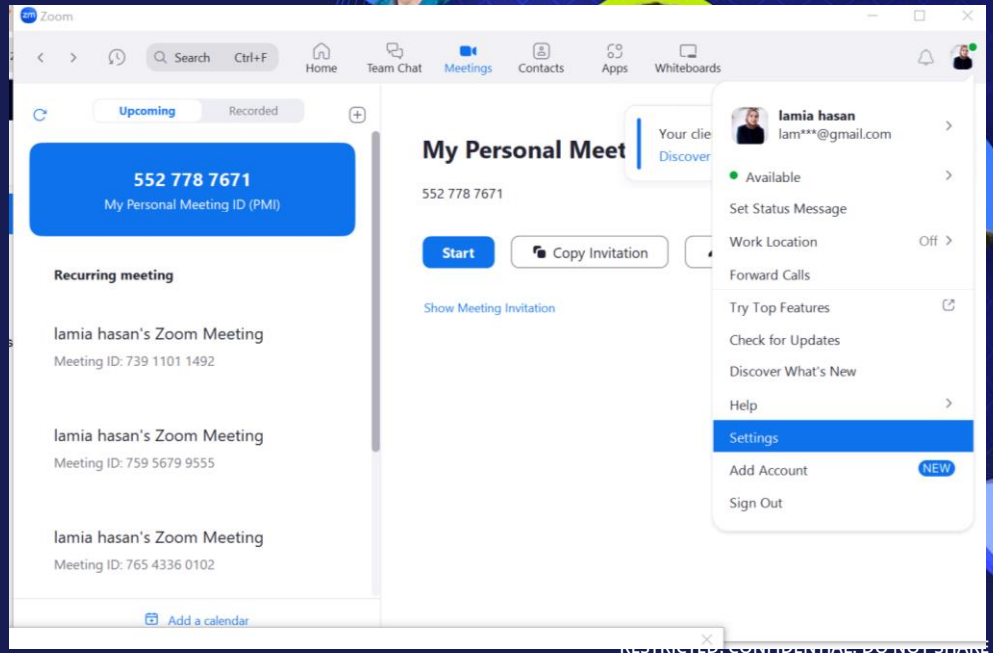
Attendance is taken automatically

Please change your name to be First Name and Last name on Zoom
Like : Lamia Zain

UDACITY

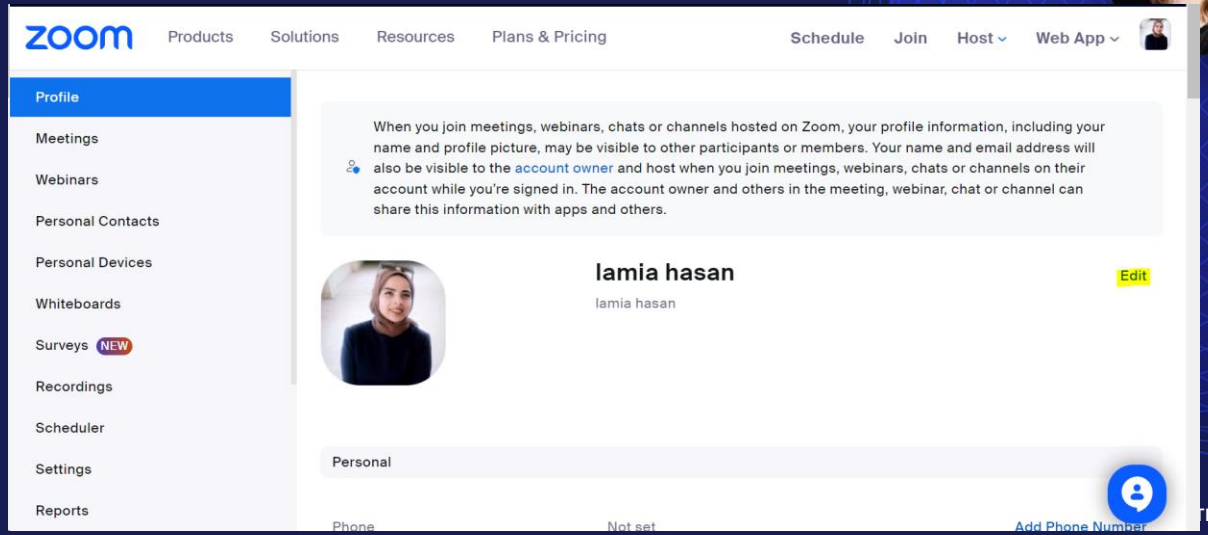
Change your Name on Zoom





UDACITY

Change your Name on Zoom



Session Lead role:

Communication Chart

Issue	Where to go?
Classroom access/ Withdrawal/ Graduation issues/ Plagiarism/ Project Review Inquiries	Email support@udacity.com
Technical Issues, Attendance, Content Related Issues/ Project inquiries	Session Lead
Session Switch/ Community related issues	Community Moderators

Program Timeline & Learning Journey

2024

April | May | June | July | August

- **Program Kickoff** — April 10, 2024
- **Revoking Deadline** — May 12, 2024
- **First Project Submission** — May 1, 2024
- **Second Project Submission** — June 5, 2024
- **Third Project Submission** — July 10, 2024
- **Fourth Project Submission** — August 21, 2024
- **End of Program** — August 28, 2024

Program Period — April 10 - August 28, 2024

Four-weeks Agenda, Weekly schedule

Week 15	Jul 17, 2024			Finish the lessons below from **the Developing your First ML Workflow** Introduction to Developing ML Workflows **[Work on/submit the #4 project: Build a ML Workflow For Scones Unlimited On Amazon SageMaker]**	Developing your First ML Workflow Introduction to Developing ML Workflows
Week 16	Jul 24, 2024			Finish the lessons below from **the Developing your First ML Workflow** SageMaker Essentials **[Work on/submit the #4 project: Build a ML Workflow For Scones Unlimited On Amazon SageMaker]**	Developing your First ML Workflow SageMaker Essentials
Week 17	Jul 31, 2024			Finish the lessons below from **the Developing your First ML Workflow** Designing Your First Workflow **[Work on/submit the #4 project: Build a ML Workflow For Scones Unlimited On Amazon SageMaker]**	Developing your First ML Workflow Designing Your First Workflow
Week 18	Aug 7, 2024			Finish the lessons below from **the Developing your First ML Workflow** Monitoring a ML Workflow **[Work on/submit the #4 project: Build a ML Workflow For Scones Unlimited On Amazon SageMaker]**	Developing your First ML Workflow Monitoring a ML Workflow Project Walkthrough: Build a ML Workflow For Scones Unlimited On Amazon SageMaker

Four-weeks Agenda, Weekly schedule

Week 19	Aug 14, 2024	Aug 14, 2024	**Build a ML Workflow For Scones Unlimited On Amazon SageMaker**	Finish the lessons below from **the Developing your First ML Workflow** **[Work on/submit the #4 project: Build a ML Workflow For Scones Unlimited On Amazon SageMaker]**	Project Walkthrough: Build a ML Workflow For Scones Unlimited On Amazon SageMaker
Week 20	Aug 21, 2024			Prepare any questions you have about the content	Ask me Anything Session
Week 21	**Aug 28, 2024**	**(FINISH & GRADUATE)**			

Student Milestone | Revoking

REVOKING

Revoking is the process by which Udacity removes a student from a Nanodegree program.

AWS reserves the right to revoke you from the program if you do not comply with program requirements.

CRITERIA

Students can be revoked if they fail to:

- Submit Project 1
- Complete the required concepts

The date of **Revoking** for the scholarship is **May 12, 2024**

Code of Conduct | Plagiarism

BASIC RULES

- Project submissions must consist of original work
- Submitted projects will be scanned for plagiarism
- Students who are found to have plagiarised will risk their Nanodegree being revoked
- Read the honor code and the rubric carefully for all projects

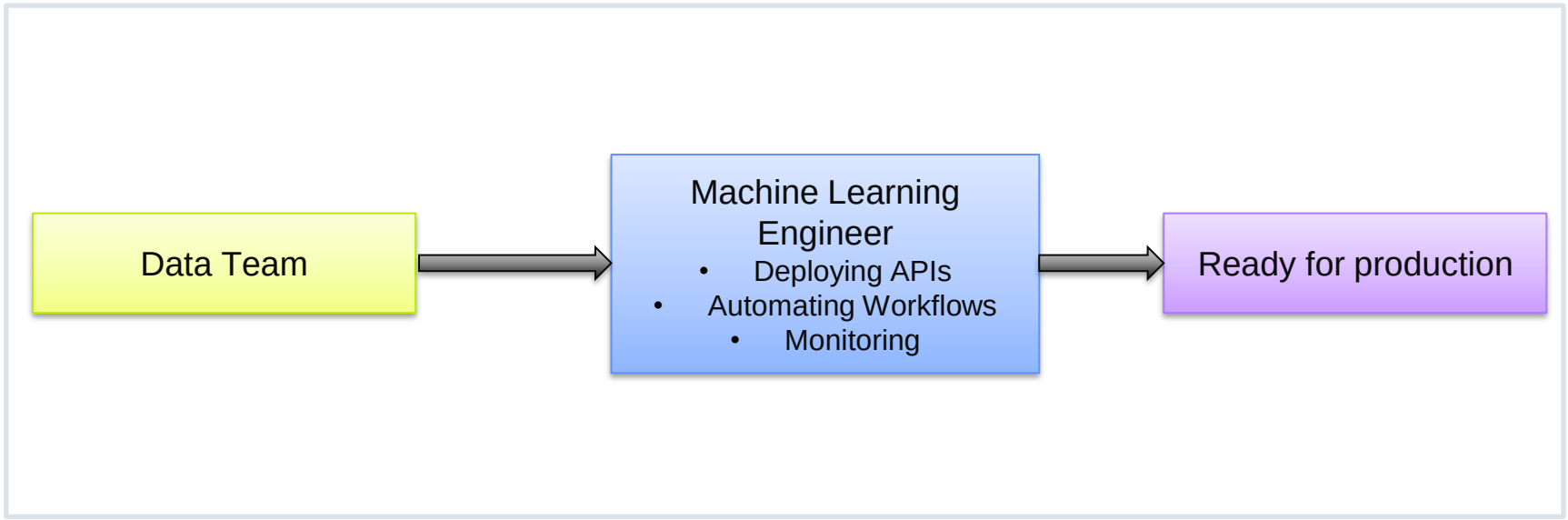
Machine Learning Engineer

Objectives:

1- Training Jobs.

2- Creating a Model.

3- Using SageMaker SDK and SageMaker Console



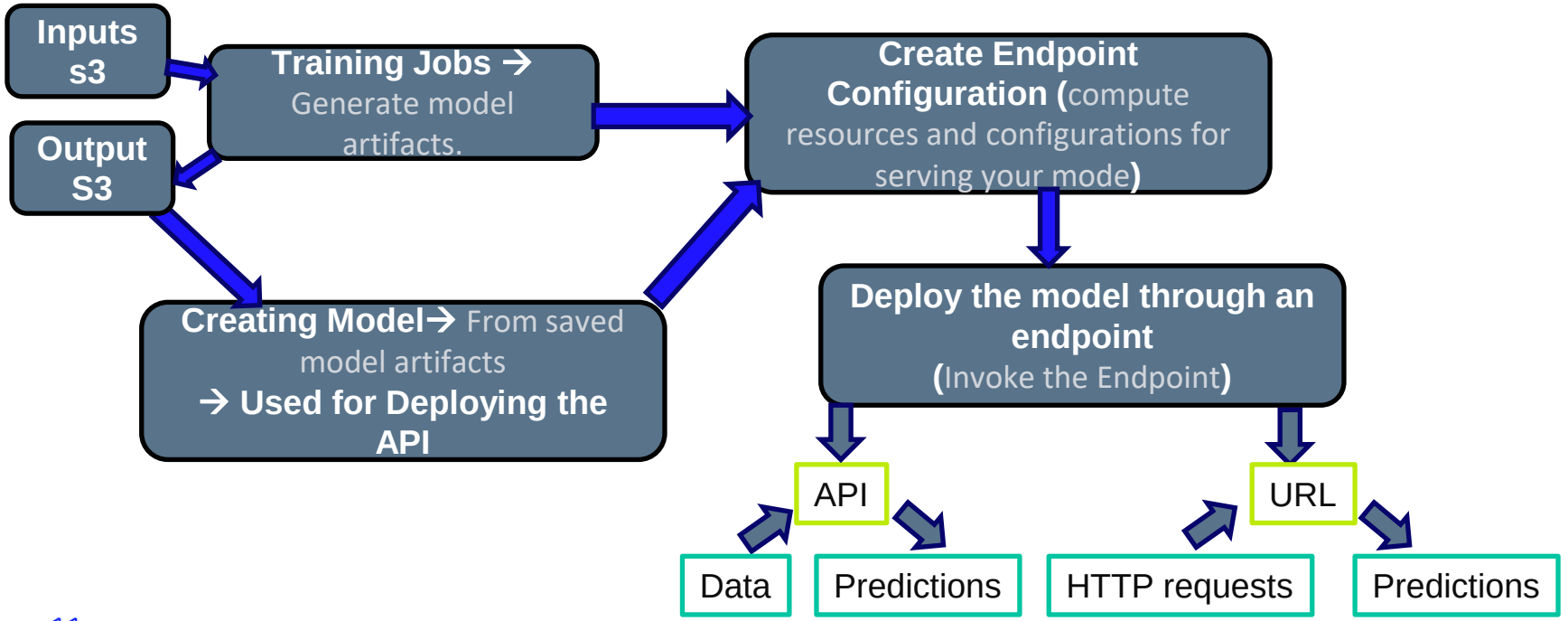
APIs: Application Programming Interface.

BOTO3 APIs:
It's an SDK for python to use and interact with all AWS services including but not limited to SageMaker.
It Gives a lower-level interface to interact with all AWS services.

Examples:
- **EC2 Instances**:
- **S3 Buckets**:
- **IAM Roles**:
- **SageMaker Services ->** Creating training jobs, deploying models, and managing endpoints.

SageMaker APIs: Designed for building, training, and deploying machine learning models.

- **Deploying APIs**: Involves making your trained machine learning models accessible so that they can receive input data and provide predictions or responses.
- SageMaker helps to achieve this by providing some services like:



Training Jobs

To construct a training Job we need:

1- Input Data Location (S3 LOCATION)
2- Output data (Model Artifacts) location (S3 Location)
3- Compute resources
4- Training Code on ECR
5- Meta data like session, role, and framework version

Your training code can be

1- An image provided by SageMaker like XGBoost
2- A script that uses popular frameworks supported by SageMaker like Scikit-Learn and PyTorch.
3- Custom Image (Copies of the computing system) → Upload to ECR → AWS will handle hardware and OS

Create a SageMaker Session: Set up the necessary SageMaker session and role.

[3]:
```
import sagemaker
import boto3
from sagemaker import get_execution_role

# Initialize a SageMaker session
sagemaker_session = sagemaker.Session()

# Get the IAM role
role = get_execution_role()
```

Prepare Data: Upload your training data to an S3 bucket.
For a training job, dataset must be uploaded to S3.

```
[ ]: # Define S3 bucket and prefix
     bucket = '<your-s3-bucket>'
     prefix = 'sagemaker/xgboost-classification'

     # Upload data to S3
     train_path = sagemaker_session.upload_data('train.csv',
                                                bucket=bucket, key_prefix=prefix + '/train')
     validation_path = sagemaker_session.upload_data('validation.csv',
                                                     bucket=bucket, key_prefix=prefix + '/validation')
```

Define the Training Job: Configure the training job with the XGBoost estimator.

```
[ ]: from sagemaker.inputs import TrainingInput
     from sagemaker.estimator import Estimator

     # Define the XGBoost container
     container = sagemaker.image_uris.retrieve('xgboost', boto3.Session().region_name, 'latest')

     # Initialize the XGBoost estimator
     xgb_estimator = Estimator(
         image_uri=container,
         role=role,
         instance_count=1,
         instance_type='ml.m5.large',
         volume_size=5,  # 5 GB
         max_run=3600,  # 1 hour
         sagemaker_session=sagemaker_session
     )

     # Set hyperparameters
     xgb_estimator.set_hyperparameters(
         objective='binary:logistic',
         num_round=100
     )
```

Launch the Training Job

```
[ ]: # Define the input data channels
     train_input = TrainingInput(s3_data=train_path, content_type='csv')
     validation_input = TrainingInput(s3_data=validation_path, content_type='csv')

[ ]: # Start the training job
     xgb_estimator.fit({'train': train_input, 'validation': validation_input})
```

Create a Model for a previously trained model

Creating a Model

```
import sagemaker
from sagemaker import get_execution_role
from sagemaker.model import Model

 # Define your SageMaker role
role = get_execution_role()

# Create a SageMaker model  Create a SageMaker model
model = Model(model_data='s3://your-bucket/model.tar.gz’,
                    role=role,
                    image_uri='your-container-image-uri’,
                    )
```

SageMaker Model

Model

class **sagemaker.model.Model**(*image_uri, model_data=None, role=None, predictor_cls=None, env=None, name=None, vpc_config=None, sagemaker_session=None, enable_network_isolation=None, model_kms_key=None, image_config=None, source_dir=None, code_location=None, entry_point=None, container_log_level=20, dependencies=None, git_config=None*)

Bases: `sagemaker.model.ModelBase`, `sagemaker.inference_recommender.inference_recommender_mixin.InferenceRecommenderMixin`

A SageMaker `Model` that can be deployed to an `Endpoint`.

Initialize an SageMaker `Model`.

Parameters:

- **image_uri** (*str or PipelineVariable*) – A Docker image URI.
- **model_data** (*str or PipelineVariable or dict*) – Location of SageMaker model data (default: None).
- **role** (*str*) – An AWS IAM role (either name or full ARN). The Amazon SageMaker training jobs and APIs that create Amazon SageMaker endpoints use this role to access training data and model artifacts. After the endpoint is created, the inference code might use the IAM role if it needs to access some AWS resources. It can be null if this is being used to create a Model to pass to a `PipelineModel` which has its own Role field. (default: None)
- **predictor_cls** (*callable[string, sagemaker.session.Session]*) – A function to call to

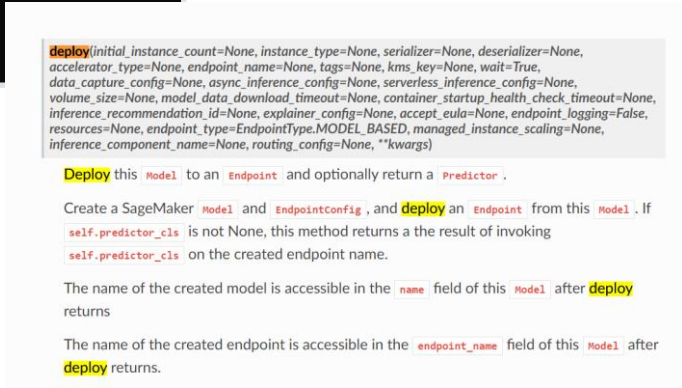
Creating an endpoint

Configure an endpoint

```
# Deploy the model as an endpoint
predictor = model.deploy(endpoint_name='your-endpoint-name',
                         initial_instance_count=1,
                         instance_type='ml.m5.large')

# Invoke the API
result = predictor.predict(data)

# Send data to the endpoint for prediction # Delete the endpoint when done
predictor.delete_endpoint()
```



deploy(*initial_instance_count=None, instance_type=None, serializer=None, deserializer=None, accelerator_type=None, endpoint_name=None, tags=None, kms_key=None, wait=True, data_capture_config=None, async_inference_config=None, serverless_inference_config=None, volume_size=None, model_data_download_timeout=None, container_startup_health_check_timeout=None, inference_recommendation_id=None, explainer_config=None, accept_eula=None, endpoint_logging=False, resources=None, endpoint_type=EndpointType.MODEL_BASED, managed_instance_scaling=None, inference_component_name=None, routing_config=None, **kwargs*)

Deploy this `Model` to an `Endpoint` and optionally return a `Predictor`.

Create a SageMaker `Model` and `EndpointConfig`, and deploy an `Endpoint` from this `Model`. If `self.predictor_cls` is not None, this method returns a the result of invoking `self.predictor_cls` on the created endpoint name.

The name of the created model is accessible in the `name` field of this `Model` after deploy returns

The name of the created endpoint is accessible in the `endpoint_name` field of this `Model` after deploy returns.

Why would you use SageMaker Services for ML Automation?

Processes are serial,
Eg. the output of training jobs is meant to be the inputs into endpoints and batch transform jobs

Processes are very similar.

In-house data (S3 Bucket).

ML libraries included and Managed by SageMaker (SKLearn and PyTorch).

Computational resources (instances for different services) accessed by SageMkaer

Many Algorithms like XGBoost are provided managed by SageMaker

Break (10 minutes)

Satisfaction Survey

Breast Cancer Dataset

XGBoost built in with SageMaker SDKs

Any Question?

Thank you

UDACITY