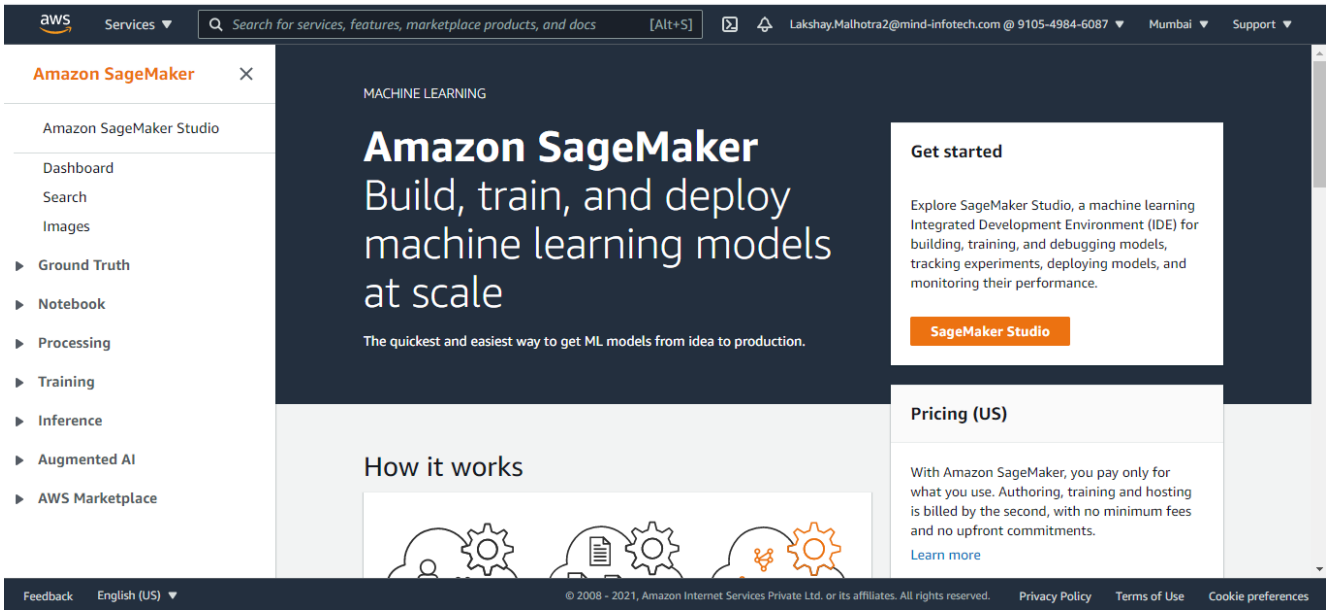
Step 1: Create an Amazon SageMaker Notebook Instance

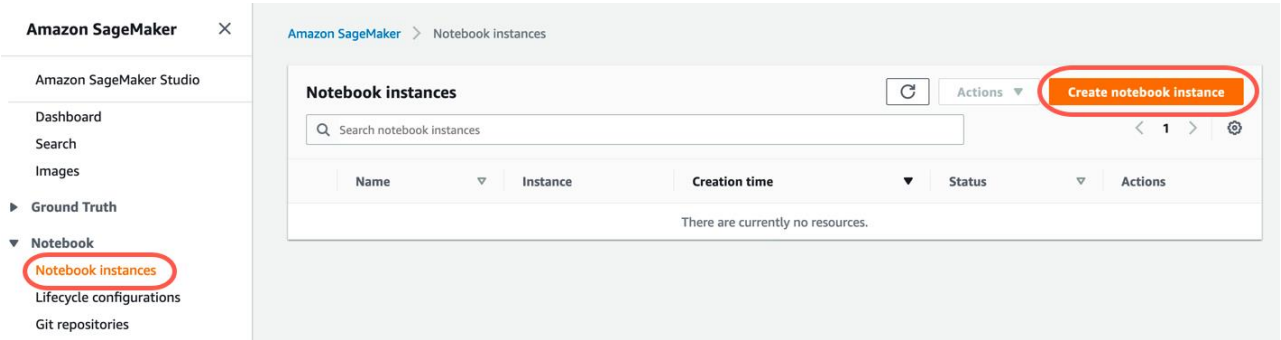
An Amazon SageMaker notebook instance is a fully managed machine learning (ML) Amazon Elastic Compute Cloud (Amazon EC2) compute instance that runs the Jupyter Notebook App. You use the notebook instance to create and manage Jupyter notebooks for preprocessing data and to train and deploy machine learning models.

To create a SageMaker notebook instance

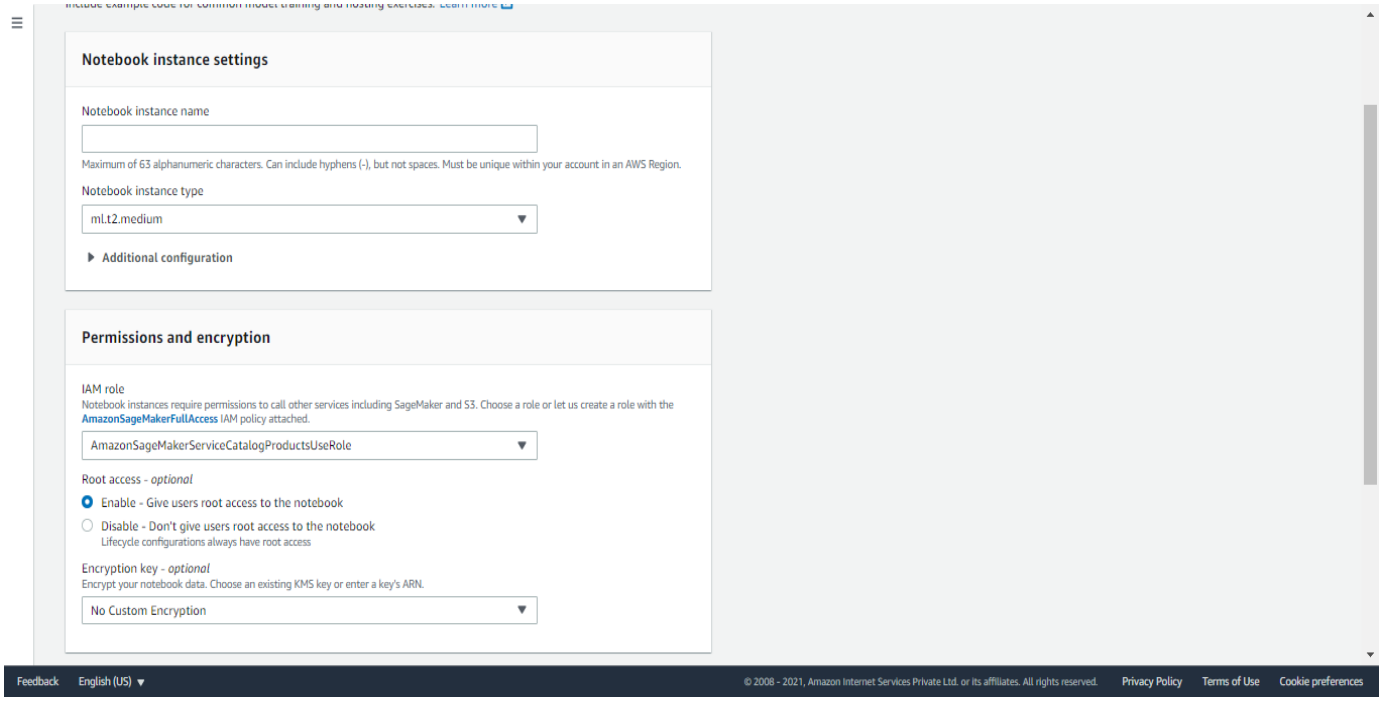
1. Open the Amazon SageMaker console at https://console.aws.amazon.com/sagemaker/



2. Choose Notebook instances, and then choose Create notebook instance.



3. On the Create notebook instance page, provide the following information (if a field is not mentioned, leave the default values):
    a. For Notebook instance name, type a name for your notebook instance.
    b. For Instance type, choose ml.t2.medium. This is the least expensive instance type that notebook instances support, and it suffices for this exercise. If a ml.t2.medium instance type isn't available in your current AWS Region, choose ml.t3.medium.
    c. For IAM role, choose Create a new role, and then choose Create role. This IAM role automatically gets permissions to access any S3 bucket that has sagemaker in the name. It gets these permissions through the AmazonSageMakerFullAccess policy, which SageMaker attaches to the role.
    d. Choose Create notebook instance.
    In a few minutes, SageMaker launches an ML compute instance—in this case, a notebook instance—and attaches a 5 GB of Amazon EBS storage volume to it. The notebook instance has a preconfigured Jupyter notebook server, SageMaker and AWS SDK libraries, and a set of Anaconda libraries.



Step 2: Create a Jupyter Notebook

To start scripting for training and deploying your model, create a Jupyter notebook in the SageMaker notebook instance. Using the Jupyter notebook, you can conduct machine learning (ML) experiments for training and inference while accessing the SageMaker features and the AWS infrastructure.

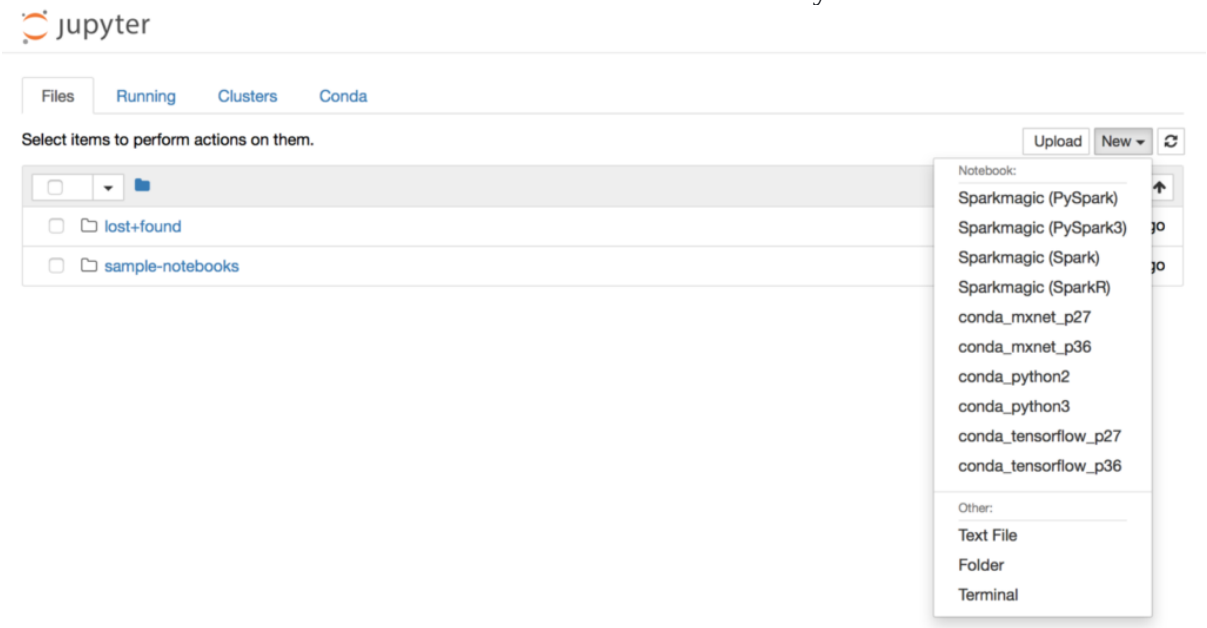
To create a Jupyter notebook

1. Open the notebook instance as follows:

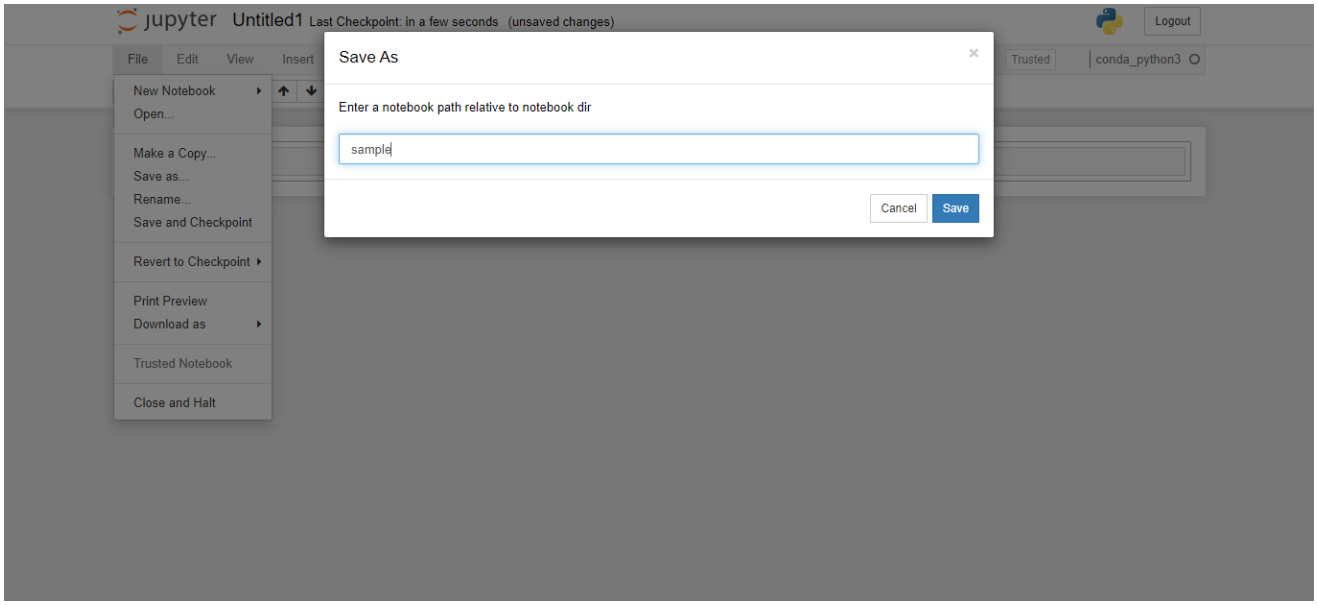
    a. Sign in to the SageMaker console at https://console.aws.amazon.com/sagemaker/

    b. On the Notebook instances page, open your notebook instance by choosing either Open JupyterLab for the JupyterLab interface or Open Jupyter for the classic Jupyter view

2. Create a notebook as follows:
    a. If you opened the notebook in the JupyterLab view, on the File menu, choose New, and then choose Notebook. For Select Kernel, choose conda_python3. This preinstalled environment includes the default Anaconda installation and Python 3.
    b. If you opened the notebook in the classic Jupyter view, on the Files tab, choose New, and then choose conda_python3. This preinstalled environment includes the default Anaconda installation and Python 3.



3. Save the notebooks as follows:
    a. In the JupyterLab view, choose File, choose Save Notebook As..., and then rename the notebook.
    b. In the Jupyter classic view, choose File, choose Save as..., and then rename the notebook.



Step 3: Train a Model

The Amazon SageMaker Python SDK provides framework estimators and generic estimators to train your model while orchestrating the machine learning (ML) lifecycle accessing the SageMaker features for training and the AWS infrastructures, such as Amazon Elastic Container Registry (Amazon ECR), Amazon Elastic Compute Cloud (Amazon EC2), Amazon Simple Storage Service (Amazon S3)

Choose the training algorithm

To choose the right algorithm for your dataset, you typically need to evaluate different models to find the most suitable models to your data. For simplicity, the SageMaker XGBoost Algorithm built-in algorithm is used throughout this tutorial without the pre-evaluation of models.

Create and Run a Training Job

After you figured out which model to use, start constructing a SageMaker estimator for training. This tutorial uses the XGBoost built-in algorithm for the SageMaker generic estimator.

To run a model training job

1. Import the Amazon SageMaker Python SDK and start by retrieving the basic information from your current SageMaker session.

```
import sagemaker

region = sagemaker.Session().boto_region_name
print("AWS Region: {}".format(region))

role = sagemaker.get_execution_role()
print("RoleArn: {}".format(role))
```

This returns the following information:

- region – The current AWS Region where the SageMaker notebook instance is running.
- role – The IAM role used by the notebook instance.

2. Create an XGBoost estimator using the sagemaker.estimator.Estimator class. In the following example code, the XGBoost estimator is named xgb_model.

```
from sagemaker.debugger import Rule, rule_configs
from sagemaker.session import TrainingInput

s3_output_location='s3://{}/{}/{}'.format(bucket, prefix, 'xgboost_model')

container=sagemaker.image_uris.retrieve("xgboost", region, "1.2-1")
print(container)

xgb_model=sagemaker.estimator.Estimator(
    image_uri=container,
    role=role,
    instance_count=1,
    instance_type='ml.m4.xlarge',
    volume_size=5,
    output_path=s3_output_location,
    sagemaker_session=sagemaker.Session(),
    rules=[Rule.sagemaker(rule_configs.create_xgboost_report())]
```

```
)
```

To construct the SageMaker estimator, specify the following parameters:

- image_uri – Specify the training container image URI. In this example, the SageMaker XGBoost training container URI is specified using sagemaker.image_uris.retrieve.
- role – The AWS Identity and Access Management (IAM) role that SageMaker uses to perform tasks on your behalf (for example, reading training results, call model artifacts from Amazon S3, and writing training results to Amazon S3).
- instance_count and instance_type – The type and number of Amazon EC2 ML compute instances to use for model training. For this training exercise, you use a single ml.m4.xlarge instance, which has 4 CPUs, 16 GB of memory, an Amazon Elastic Block Store (Amazon EBS) storage, and a high network performance
- train_volume_size – The size, in GB, of the EBS storage volume to attach to the training instance. This must be large enough to store training data if you use File mode
- output_path – The path to the S3 bucket where SageMaker stores the model artifact and training results.
- sagemaker_session – The session object that manages interactions with SageMaker API operations and other AWS service that the training job uses.
- rules – Specify a list of SageMaker Debugger built-in rules. In this example, the create_xgboost_report() rule creates an XGBoost report that provides insights into the training progress and results.

3. Set the hyperparameters for the XGBoost algorithm by calling the set_hyperparameters method of the estimator.

```
xgb_model.set_hyperparameters(
    max_depth = 5,
    eta = 0.2,
    gamma = 4,
    min_child_weight = 6,
    subsample = 0.7,
    objective = "binary:logistic",
    num_round = 1000
)
```

4. Use the TrainingInput class to configure a data input flow for training. The following example code shows how to configure TrainingInput objects to use the training and validation datasets you uploaded to Amazon S3 in the Split the Dataset into Train, Validation, and Test Datasets section.

```
from sagemaker.session import TrainingInput

train_input = TrainingInput(
    "s3://{}/{}/{}".format(bucket, prefix, "data/train.csv"), content_type="csv"
)
validation_input = TrainingInput(
    "s3://{}/{}/{}".format(bucket, prefix, "data/validation.csv"), content_type="csv"
)
```

5. To start model training, call the 1. estimator's fit method with the training and validation datasets. By setting wait=True, the fit method displays progress logs and waits until training is complete.xgb_model.fit({"train": train_input, "validation": validation_input}, wait=True)

Step 4: Deploy the Model

Deploy the Model to SageMaker Hosting Services

To host a model through Amazon EC2 using Amazon SageMaker, deploy the model that you trained in Create and Run a Training Job by calling the deploy method of the xgb_model estimator. When you call the deploy method, you must specify the number and type of EC2 ML instances that you want to use for hosting an endpoint.

```
import sagemaker
from sagemaker.serializers import CSVSerializer
xgb_predictor=xgb_model.deploy(
    initial_instance_count=1,
    instance_type='ml.t2.medium',
    serializer=CSVSerializer()
)
```

- initial_instance_count (int) – The number of instances to deploy the model.
- instance_type (str) – The type of instances that you want to operate your deployed model.
- serializer (int) – Serialize input data of various formats (a NumPy array, list, file, or buffer) to a CSV-formatted string. We use this because the XGBoost algorithm accepts input files in CSV format.

The deploy method creates a deployable model, configures the SageMaker hosting services endpoint, and launches the endpoint to host the model.

Step 6: Evaluate the Model

To evaluate the model and use it in production, invoke the endpoint with the test dataset and check whether the inferences you get returns a target accuracy you want to achieve.

To evaluate the model

1. Set up the following function to predict each line of the test set. In the following example code, the rows argument is to specify the number of lines to predict at a time. You can change the value of it to perform a batch inference that fully utilizes the instance's hardware resource.

```
import numpy as np
def predict(data, rows=1000):
    split_array = np.array_split(data, int(data.shape[0] / float(rows) + 1))
    predictions = ''
    for array in split_array:
        predictions = ','.join([predictions, xgb_predictor.predict(array).decode('utf-8')])
    return np.fromstring(predictions[1:], sep=',')
```

2. Run the following code to make predictions of the test dataset
   `predictions=predict(test.to_numpy()[:,1:])`

Step 7: Clean Up

To avoid incurring unnecessary charges, use the AWS Management Console to delete the endpoints and resources that you created while running the exercises.

1. Open the Amazon SageMaker console at https://console.aws.amazon.com/sagemaker/ and delete the following resources:

- The endpoint. Deleting the endpoint also deletes the ML compute instance or instances that support it.
  1. Under Inference, choose Endpoints.
  2. Choose the endpoint that you created in the example, choose Actions, and then choose Delete.
- The endpoint configuration.

1. Under Inference, choose Endpoint configurations.
2. Choose the endpoint configuration that you created in the example, choose Actions, and then choose Delete.

- The model.
    1. Under Inference, choose Models.
    2. Choose the model that you created in the example, choose Actions, and then choose Delete.
- The notebook instances. Before deleting the notebook instance, stop it.
    1. Under Notebook, choose Notebook instances.
    2. Choose the notebook instance that you created in the example, choose Actions, and then choose Stop. The notebook instance takes several minutes to stop. When the Status changes to Stopped, move on to the next step.
    3. Choose Actions, and then choose Delete.

2. Open the Amazon S3 console at https://console.aws.amazon.com/s3/, and then delete the bucket that you created for storing model artifacts and the training dataset.

3. Open the Amazon CloudWatch console at https://console.aws.amazon.com/cloudwatch/, and then delete all of the log groups that have names starting with /aws/sagemaker/.

Reference for Hands-on:

Creating a scikit-learn Random Forest Classifier in AWS SageMaker - A Cloud Guru