FRANKFURT UNIVERSITY OF APPLIED SCIENCES

High Integrity Systems

Cloud Computing

Machine Learning Model Computation in AWS and Azure

Melisa Xhepa (1378689)
Navya Sree Kanakala (1381948)

UNDER THE GUIDANCE OF

Prof. Dr. Christian Baun

July 8, 2022

1.5

Contents

1 Introduction

Machine learning (ML) is the study and development of algorithms that get better at a particular task as they learn more about it. Through the use of machine learning, systems can learn from their past performance and become better at what they do [1]. Machine learning, like the human brain, relies on input, such as training data or knowledge graphs, to understand entities, domains, and the relationships between them. Deep learning can begin once entities have been defined.

Most machine learning models start with training data, such as an enormous folder of images, which the machine processes and starts to ”understand” statistically. Since there is no active human programming involved in machine learning, the machine algorithm itself develops and changes as it interacts with the training data [1]. The algorithm eventually becomes prepared for ”real” data. As it processes fresh data, it keeps changing and eventually comes up with a response or fix on its own. The fact that not all problems are composed of a closed set of variables and procedures makes machine learning crucial. Traditional algorithms deal with bounded, highly specific problems. By using machine learning, we can teach a model to solve problems that we may not have known how to initially.

The distinction between algorithms and training must also be kept in mind. Depending on the use case, one may look for a model using the technique and supply new data to train it, or one may look for a model that has already been trained to handle the problem you're attempting to solve.

We may utilize machine learning techniques, which are quite varied and have a large variety of models, to handle problems like result prediction, estimate (when a value changes due to many natural causes), and element classification.

The following are some examples of machine learning algorithms:

- **Anomaly detection:** This can reveal uncommon occurrences, events, or observations that stand out significantly from the bulk of the data.
- **Classification:** The data points provided can be classified effectively using this. It can be used, for example, to locate tumors or find spam.
- **Clustering:** The objects in the same group are more similar to one another than to those in other groups thanks to this method of grouping a collection of objects. Pattern recognition, image analysis, data compression, biological classification, and insurance are a few examples where it is helpful.
- **Regression:** With the help of a set of data points, this is used to determine the quantity that can be expected as an input. The best coefficients of this mathematical relationship can be inferred under the assumption that inputs and outputs are related mathematically via a linear, polinomic, logistic, or other relationship. This kind of analysis is used to demonstrate a mathematical relationship or correlation in any type of experiment that collects data.
- **Computer vision:** It is possible to read text from handwritten notes or images, identify people or familiar landmarks, and analyze videos in real time.

All of these machine learning techniques are linked by the fact that they all rely on automatically learning from a large amount of data. The algorithm architecture and a few initial parameters can be defined by the programmer, and after that, the program learns by analyzing the data.

2 Machine Learning in the Cloud

Each day, more and more applications for artificial intelligence and machine learning are being developed. Also probable is the fact that a sizeable portion of it will take place in the cloud. Over the years, major reorganizations that strategically integrated AI into their organizational structures were carried out by leading cloud computing platforms like Amazon, Google, and Microsoft.

How do you choose the best platform for your machine learning projects if the cloud is where they will be deployed? In this project, we will examine the machine learning image classification service as an example, provided by both Microsoft Azure, and Amazon Web Services [2]. To create a machine learning solution, you are not required to use a cloud service provider. After all, businesses can run a variety of open source machine learning frameworks on their own hardware.

There are many advantages to moving some or all of your machine learning projects to the cloud. The pay-per-use cloud model is ideal for bursting AI or machine learning workloads, and you can use the speed and power of GPUs for training without having to make a hardware investment. Additionally, the cloud enables companies to rapidly scale up their machine learning experiments as projects go into production and demand for those features increases. What's perhaps more significant is that the cloud enables intelligent capabilities without the requirement for highly specialized knowledge in data science or artificial intelligence, which are both uncommon and in short supply [2].

There are numerous ways to integrate intelligent features into enterprise applications using AWS, Microsoft Azure, and Google Cloud Platform that don't necessitate a team of data scientists or deep knowledge of AI or machine learning theory.

While providing superior features from scalability to security, the hypercloud providers (AWS, Azure, and GCP) are able to offer a lower total cost of ownership. Given that there are a lot of cloud-based models available, using pre-made models actually is more beneficial. Because of the ever-increasing number of algorithm optimizations for supporting parallel computing, utilizing less memory, and quickly starting up, running, and shutting down, there is frequently no need to reinvent the wheel in machine learning. There will always be a quicker execution time if we use pre-written models and run an experiment in the cloud. This is because the models have already been optimized and refined with the best training available for their task by experts, and the computing power a cloud provider can deliver produces better performance than what we can achieve with on-premises computing.

3 Use Case and Motivation

The air sacs in one or both lungs become inflamed when a person has pneumonia, or a lung infection. Symptoms and a physical exam are frequently used to make a diagnosis. To confirm the diagnosis, chest X-rays may be required.
Our goal was to create a transfer learning algorithm that would efficiently process medical images and quickly and accurately diagnose any key pathology present in each image. In this project, we establish a diagnostic tool based on a deep-learning framework for the screening and diagnosing pediatric pneumonia. Our framework makes use of transfer learning, which requires a lot less data to train a neural network than traditional methods. In the long run, this tool might help hasten the diagnosis and referral of these treatable conditions, enabling earlier treatment and better clinical outcomes.

3.1 Content

There are 5,856 validated chest X-ray images in this dataset. Two sets of independent patients' images are used for the training and testing. Before training, Each image imported into the database started with a label matching the most recent diagnosis of the patient. The following labels are placed on images:

(disease: NORMAL/BACTERIA/VIRUS) - (randomized patient ID) - (image number of a patient).

4 Technologies Used

4.1 Amazon SageMaker

Machine learning models can be quickly and easily built, trained, and deployed at any scale using Amazon SageMaker, a fully-managed service for data scientists and developers. For the development, improvement, and deployment of machine learning models, Amazon SageMaker includes modules that can be used jointly or separately.
Amazon SageMaker makes it simple to build ML models by providing everything you need to quickly connect to your training data, choose and optimize the best algorithm and framework for your application, and create ML models that are ready for training [3]. Using hosted Jupyter notebooks, a feature of Amazon SageMaker, you can easily explore and visualize your training data.

4.2 S3 Bucket

The Amazon Simple Storage Service for object storage has scalability, data accessibility, security, and performance (Amazon S3). Customers from all sizes and industries can use Amazon S3 to store and safeguard any amount of data for a variety of use cases, such as data lakes, websites, mobile applications, backup and restore, archives, enterprise applications, IoT devices, and big data analytics. You can optimize, arrange, and set up the access to your data using its management features [4]. We are not using an S3 bucket in this project because we are implementing our model in Jupyter Notebook and the data is being read directly there. The bucket is created as a result of the IAM role and Sagemaker Notebook Instance creation.

4.3 Amazon EC2

Elastic Compute Cloud (EC2) is an on-demand computing service on the AWS cloud platform. Computing encompasses all of the services that a computing device can provide, as well as the flexibility of a virtual environment. It also allows users to configure their instances according to their needs, such as allocating RAM, ROM, and storage based on the needs of the current task. Rather than buying your own hardware and connecting it to a network, Amazon provides you with nearly limitless virtual machines to run your applications while they manage the hardware. Auto-Scaling Groups are a unique feature of EC2 instances that is essential to cloud computing. This enables EC2 instances to dynamically add additional computing power as demand exceeds certain thresholds, such as CPU utilization.

4.4 IAM

The IAM service manages who can access resources after being authenticated (signed in) and authorized (having permissions). You start off with a single sign-in identity when you first create an AWS account, and this identity has full access [5] to all of the account's resources and AWS services. But if you're not the root user (for instance, in a company's AWS account), the admin user might have placed some restrictions on what services you can use.

4.5 Jupyter Notebook

Data scientists can create and share documents with live code, equations, computational output, visualizations, and other multimedia resources as well as illustrative text using the free and open-source web application known as Jupyter Notebook. Data cleaning and transformation, numerical simulation, exploratory data analysis, data visualization, statistical modeling, machine learning, deep learning, and many other data science tasks can all be performed [6] using Jupyter Notebooks.
A Jupyter notebook is composed of two parts: a front-end web page and a back-end kernel. Data scientists can enter programming code or text into rectangular "cells" on the front-end web page. The code is then passed to the back-end kernel, which executes it and returns the results.

4.6 AzureML

AzureML is an Azure cloud-based machine learning service used for the development, training, and deployment of Machine Learning models and solutions. AzureML provides a variety of features for processing large amounts of data [7] in the cloud, developing machine learning models and training, and finally deploying models as web services. It can manage datasets, training statistics, and multiple trained models, which can then be tested.

4.7 Azure Blob Storage

Azure Blob Storage is the cloud's object storage service. Unstructured data, such as objects, don't need a particular data model. In order to store files, Azure Blob Storage saves the objects using the flat namespace. The name of the stored item is the key, and its data is the value, of the key-value pair [7] in which objects are saved.

4.8 Azure File Storage

Azure File Storage is the cloud file storage solution. Before configuring it, it should be mounted or deployed on any operating system. Furthermore, because it is similar to a regular file system in Windows operating systems, users are familiar with this type of storage. The only distinction is that Azure File Storage is remotely mounted and has an unlimited storage capacity [7].

4.9 Azure Machine Learning Workspace

The Azure Machine Learning workspace provides a centralized location for working with all of the artifacts generated by Azure Machine Learning. The workspace archives all training runs, including logs, metrics, output, and a snapshot of your scripts. This data is used to determine which training run yields the best model. You can use an Azure Machine Learning workspace to manage data, resources, and other aspects of your machine learning workloads if you add it to your Azure subscription.

4.10 Azure Compute

A compute instance is a cloud-based workstation that is completely managed and tailored for your machine learning development environment. The Azure Machine Learning compute instance allows you to write, train, and deploy models in your workspace using a fully integrated notebook experience. Jupyter notebooks may be executed in VS Code utilizing a compute instance as the remote server without the requirement for SSH. You may also integrate VS Code through a remote SSH plugin. You may add kernels and install software to your compute instance.

5 Architecture

5.1 AWS

Figure 1 depicts the architecture that can be produced by following the document's deployment instructions. To build a machine learning model on AWS SageMaker, one IAM Role and one instance are used. The instance is Amazon EC2, and the EC2 instance will operate within AWS SageMaker. The model will be trained using Jupyter Notebook, an AWS SageMaker built-in function. The trained model will be saved on the Jupyter Notebook's home page. The deployed machine learning model is adaptable and can run on any cloud or local system.

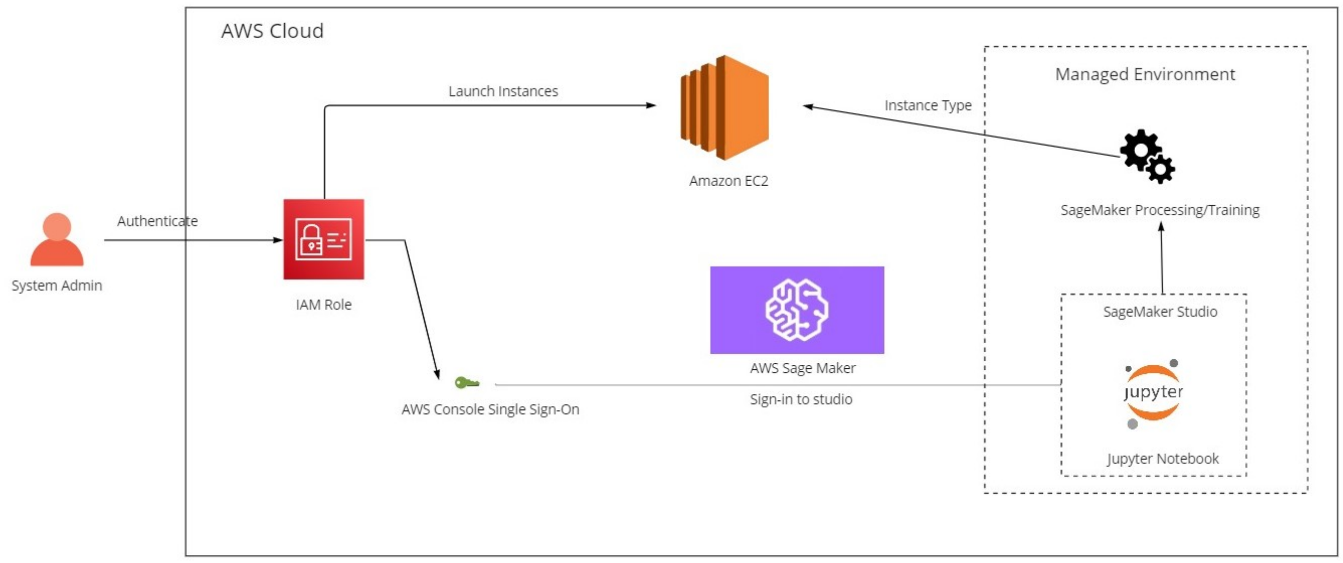


Figure 1: AWS Architecture

AWS Flowchart

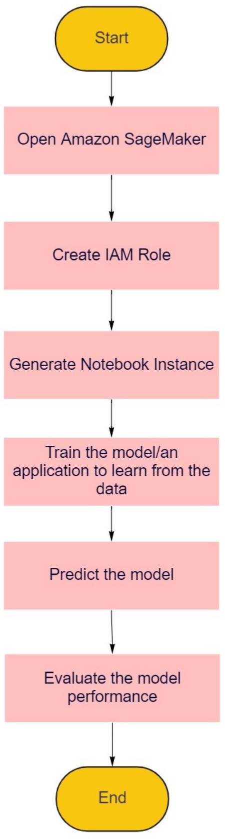

Start
Open Amazon SageMaker
Create IAM Role
Generate Notebook Instance
Train the model/an application to learn from the data
Predict the model
Evaluate the model performance
End

Figure 2: AWS Flowchart

5.2 Azure

Figure 3 depicts the architecture that can be produced by following the document's deployment instructions. This project makes use of chest X-ray pictures, but the data can be of any kind, such as a data lake or blob storage, as long as it is supported by Azure. The input photographs are uploaded to Azure Blob storage as the data asset, and an abstract representation of the data is constructed in Azure ML Dataset. In a Python Jupyter Notebook, we employ the Azure Machine Learning training and deployment workflow. Then, using your data, notebook is used to build our own machine learning model. A Compute VM Instance is generated in the Model Training, which also serves as a training module for the image classification algorithm. This module may track all experiments, as well as their associated runs, output, metrics, logs, and model registry. Following deployment, a REST endpoint is built, which can be used to carry out the classification job. The machine learning model that has been deployed is adaptable and can be accessed by any web application.

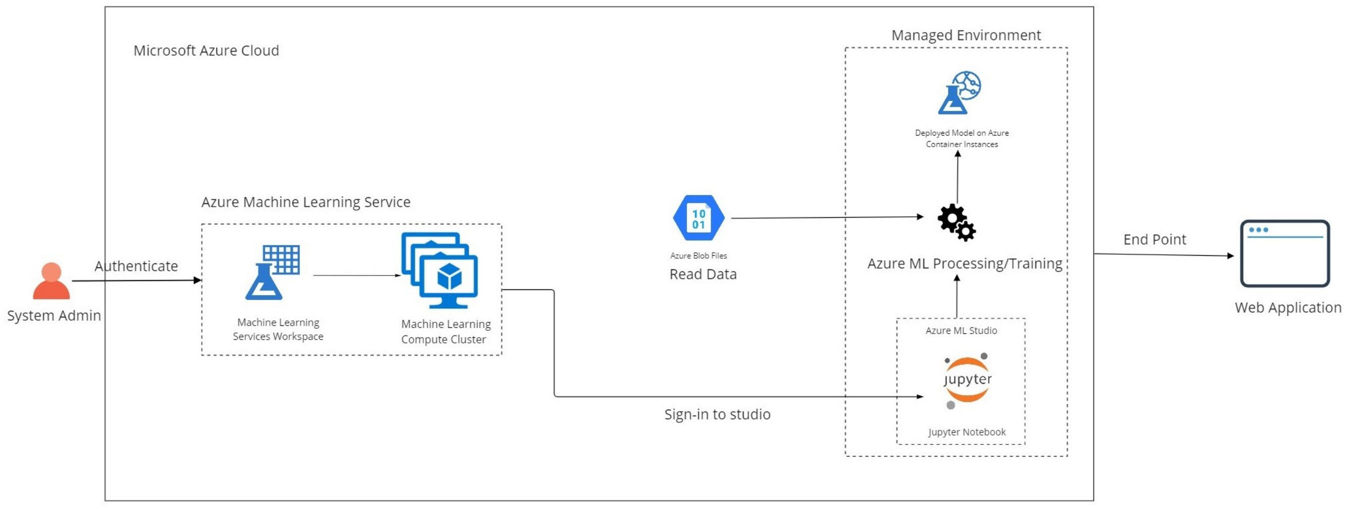

Microsoft Azure Cloud
Managed Environment
Deployed Model on Azure Container Instances
Azure Machine Learning Service
Authenticate
System Admin
Machine Learning Services Workspace
Machine Learning Compute Cluster
Azure Blob Files
Read Data
Azure ML Processing/Training
End Point
Web Application
Azure ML Studio
jupyter
Jupyter Notebook
Sign-in to studio

Figure 3: Azure Architecture

Azure Flowchart

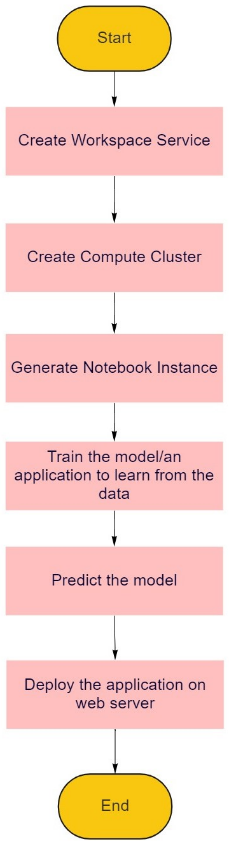

Start
Create Workspace Service
Create Compute Cluster
Generate Notebook Instance
Train the model/an application to learn from the data
Predict the model
Deploy the application on web server
End

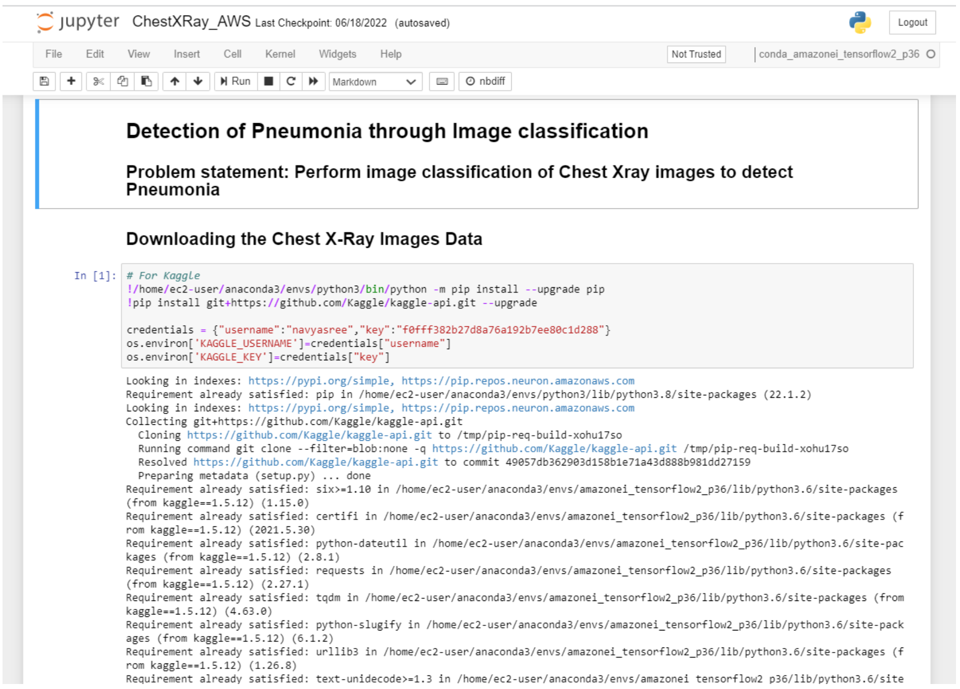
Figure 4: Azure Flowchart

6 Implementation

This section will give a closer look to the image classification implementation on Azure and AWS.

6.1 Generate Example Data for The Implementation

Training a model requires example data. The problem we are trying to solve with the model will determine the kind of data needed. In our case we want to create a model that can diagnose pneumonia using a chest X-Ray as the input. To train such a model, we'll need examples of chest X-Rays that show this.
Both the input values and the output response that you want the model to learn must be included in the training set of data. It is critical to divide the dataset into training, and test sets when you are working with it [8].


jupyter ChestXRay_AWS Last Checkpoint: 06/18/2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
conda_amazonei_tensorflow2_p36
Run
Markdown
nbdiff

Detection of Pneumonia through Image classification

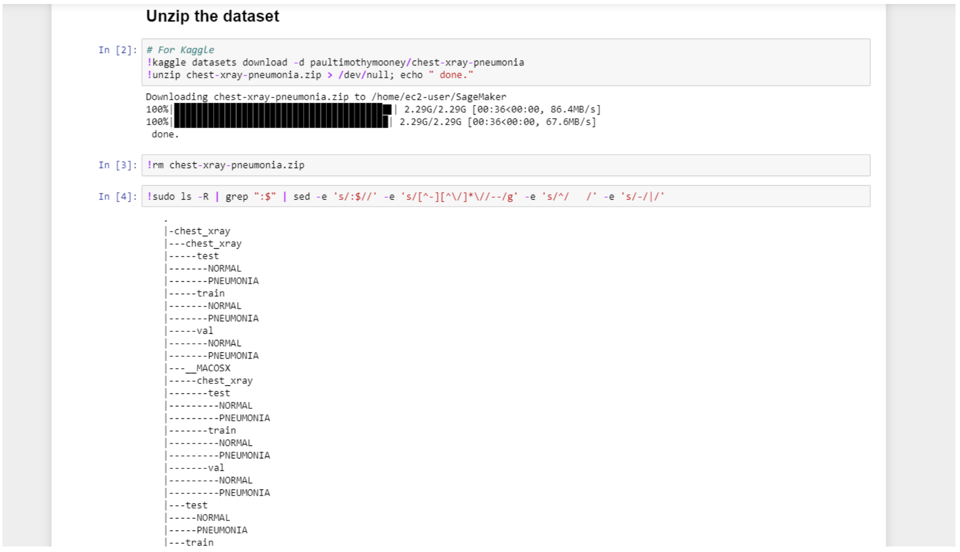
Problem statement: Perform image classification of Chest Xray images to detect Pneumonia

Downloading the Chest X-Ray Images Data

```
In [1]: # For Kaggle
!/home/ec2-user/anaconda3/envs/python3/bin/python -m pip install --upgrade pip
!pip install git+https://github.com/Kaggle/kaggle-api.git --upgrade

credentials = {"username":"navyasree","key":"f0fff382b27d8a76a192b7ee80c1d288"}
os.environ['KAGGLE_USERNAME']=credentials["username"]
os.environ['KAGGLE_KEY']=credentials["key"]
```

```
Looking in indexes: https://pypi.org/simple, https://pip.repos.neuron.amazonaws.com
Requirement already satisfied: pip in /home/ec2-user/anaconda3/envs/python3/lib/python3.8/site-packages (22.1.2)
Looking in indexes: https://pypi.org/simple, https://pip.repos.neuron.amazonaws.com
Collecting git+https://github.com/Kaggle/kaggle-api.git
  Cloning https://github.com/Kaggle/kaggle-api.git to /tmp/pip-req-build-xohu17so
  Running command git clone --filter=blob:none -q https://github.com/Kaggle/kaggle-api.git /tmp/pip-req-build-xohu17so
  Resolved https://github.com/Kaggle/kaggle-api.git to commit 49057db362903d158b1e71a43d888b981dd27159
  Preparing metadata (setup.py) ... done
Requirement already satisfied: six>=1.10 in /home/ec2-user/anaconda3/envs/amazonei_tensorflow2_p36/lib/python3.6/site-packages
(from kaggle==1.5.12) (1.15.0)
Requirement already satisfied: certifi in /home/ec2-user/anaconda3/envs/amazonei_tensorflow2_p36/lib/python3.6/site-packages (f
rom kaggle==1.5.12) (2021.5.30)
Requirement already satisfied: python-dateutil in /home/ec2-user/anaconda3/envs/amazonei_tensorflow2_p36/lib/python3.6/site-pac
kages (from kaggle==1.5.12) (2.8.1)
Requirement already satisfied: requests in /home/ec2-user/anaconda3/envs/amazonei_tensorflow2_p36/lib/python3.6/site-packages
(from kaggle==1.5.12) (2.27.1)
Requirement already satisfied: tqdm in /home/ec2-user/anaconda3/envs/amazonei_tensorflow2_p36/lib/python3.6/site-packages (from
kaggle==1.5.12) (4.63.0)
Requirement already satisfied: python-slugify in /home/ec2-user/anaconda3/envs/amazonei_tensorflow2_p36/lib/python3.6/site-pack
ages (from kaggle==1.5.12) (6.1.2)
Requirement already satisfied: urllib3 in /home/ec2-user/anaconda3/envs/amazonei_tensorflow2_p36/lib/python3.6/site-packages (f
rom kaggle==1.5.12) (1.26.8)
Requirement already satisfied: text-unidecode>=1.3 in /home/ec2-user/anaconda3/envs/amazonei tensorflow2 p36/lib/python3.6/site
```

Figure 5: Image Processing on AWS



Unzip the dataset

```
In [2]: # For Kaggle
!kaggle datasets download -d paultimothymooney/chest-xray-pneumonia
!unzip chest-xray-pneumonia.zip > /dev/null; echo " done."
```

```
Downloading chest-xray-pneumonia.zip to /home/ec2-user/SageMaker
100%|█████████████████████████████████████| 2.29G/2.29G [00:36<00:00, 86.4MB/s]
100%|█████████████████████████████████████| 2.29G/2.29G [00:36<00:00, 67.6MB/s]
 done.
```

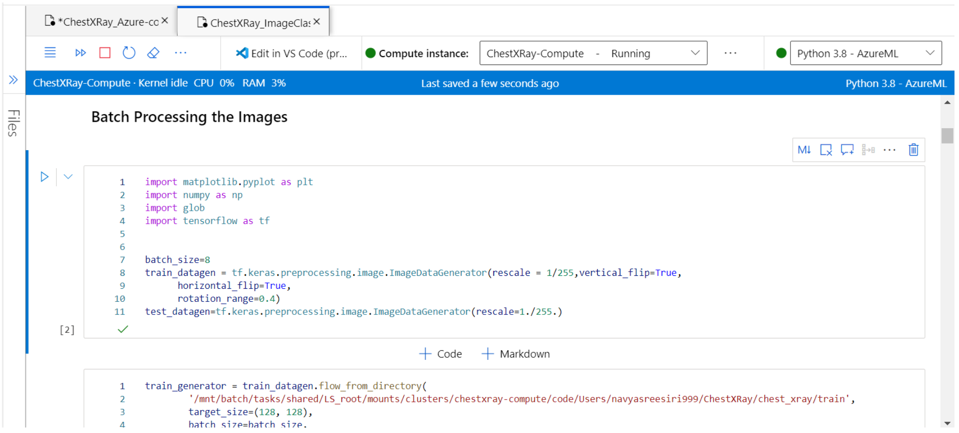
```
In [3]: !rm chest-xray-pneumonia.zip
```

```
In [4]: !sudo ls -R | grep ":$" | sed -e 's/:$//' -e 's/[^-][^\/]*\//--/g' -e 's/^/   /' -e 's/-/|/'
```

```
   .
   |-chest_xray
   |---chest_xray
   |-----test
   |-------NORMAL
   |-------PNEUMONIA
   |-----train
   |-------NORMAL
   |-------PNEUMONIA
   |-----val
   |-------NORMAL
   |-------PNEUMONIA
   |---__MACOSX
   |-----chest_xray
   |-------test
   |---------NORMAL
   |---------PNEUMONIA
   |-------train
   |---------NORMAL
   |---------PNEUMONIA
   |-------val
   |---------NORMAL
   |---------PNEUMONIA
   |---test
   |-----NORMAL
   |-----PNEUMONIA
   |---train
```

Figure 6: Image Processing

The learning process uses only the first one as the model's training data source. In order to confirm that the model is capable of making meaningful predictions, you then use the validation set, which the model has not seen. In order to compare the output that the validation data must produce with what the model inferred from it, you process the validation data using the trained model. The output should be very accurate if the model training process went well.

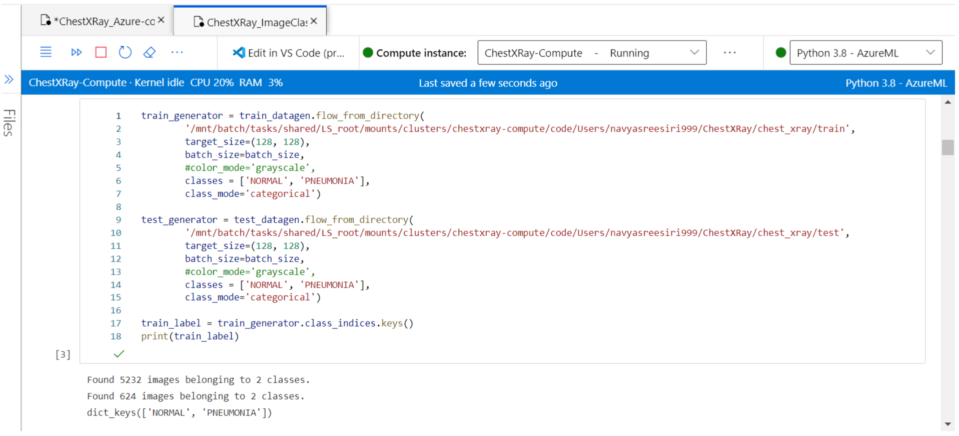

*ChestXRay_Azure-co
ChestXRay_ImageCla:
Edit in VS Code (pr...
Compute instance:
ChestXRay-Compute - Running
Python 3.8 - AzureML
ChestXRay-Compute · Kernel idle CPU 0% RAM 3%
Last saved a few seconds ago
Files
Batch Processing the Images
Code
Markdown

```
import matplotlib.pyplot as plt
import numpy as np
import glob
import tensorflow as tf


batch_size=8
train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale = 1/255,vertical_flip=True,
      horizontal_flip=True,
      rotation_range=0.4)
test_datagen=tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255.)
```

[2]

```
train_generator = train_datagen.flow_from_directory(
        '/mnt/batch/tasks/shared/LS_root/mounts/clusters/chestxray-compute/code/Users/navyasreesiri999/ChestXRay/chest_xray/train',
        target_size=(128, 128),
        batch_size=batch_size,
```

Figure 7: Image Processing on Azure



```
train_generator = train_datagen.flow_from_directory(
        '/mnt/batch/tasks/shared/LS_root/mounts/clusters/chestxray-compute/code/Users/navyasreesiri999/ChestXRay/chest_xray/train',
        target_size=(128, 128),
        batch_size=batch_size,
        #color_mode='grayscale',
        classes = ['NORMAL', 'PNEUMONIA'],
        class_mode='categorical')

test_generator = test_datagen.flow_from_directory(
        '/mnt/batch/tasks/shared/LS_root/mounts/clusters/chestxray-compute/code/Users/navyasreesiri999/ChestXRay/chest_xray/test',
        target_size=(128, 128),
        batch_size=batch_size,
        #color_mode='grayscale',
        classes = ['NORMAL', 'PNEUMONIA'],
        class_mode='categorical')

train_label = train_generator.class_indices.keys()
print(train_label)
```

[3]

```
Found 5232 images belonging to 2 classes.
Found 624 images belonging to 2 classes.
dict_keys(['NORMAL', 'PNEUMONIA'])
```

Figure 8: Image Processing on Azure

The following is done to get data ready for analysis:

1. Fetch the data — One may use publicly accessible datasets or internal sample data repositories. The datasets are typically combined into a single repository.

2. Clean the data — Examine the data and clean it as necessary for more effective model training.

3. Prepare or transform the data—We may need to execute extra data transformations to increase performance. We may, for example, opt to mix qualities. Instead of utilizing temperature and humidity variables separately, we may combine them into a single attribute to achieve a better model if the model anticipates circumstances that necessitate deicing an airplane.

4. Data preparation or transformation — To improve performance, we might need to carry out additional data transformations.

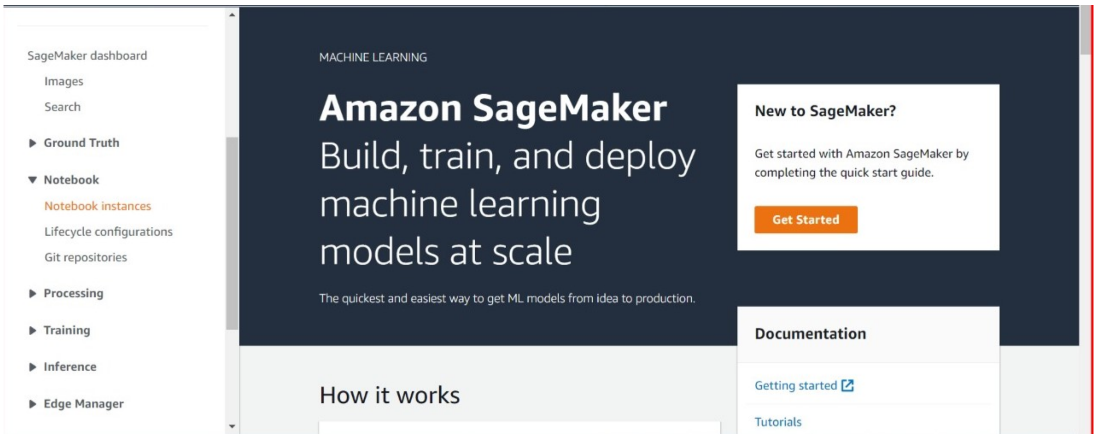
We pre-process sample data in our notebook instance's notebook. The dataset is retrieved, explored, and ready for model training using the notebook. This applies to both of our implementations.

6.2 AWS Implementation

This section demonstrates how to use Amazon Sagemaker Image classification algorithm to train a dataset.

6.2.1 Create an account

On the AWS Free tier website, users should register for a free tier account. When you register for Amazon Web Services (AWS), all of the AWS services, including Amazon SageMaker, are immediately added to your AWS account.



Amazon SageMaker
Build, train, and deploy machine learning models at scale

The quickest and easiest way to get ML models from idea to production.

New to SageMaker?

Get started with Amazon SageMaker by completing the quick start guide.

How it works

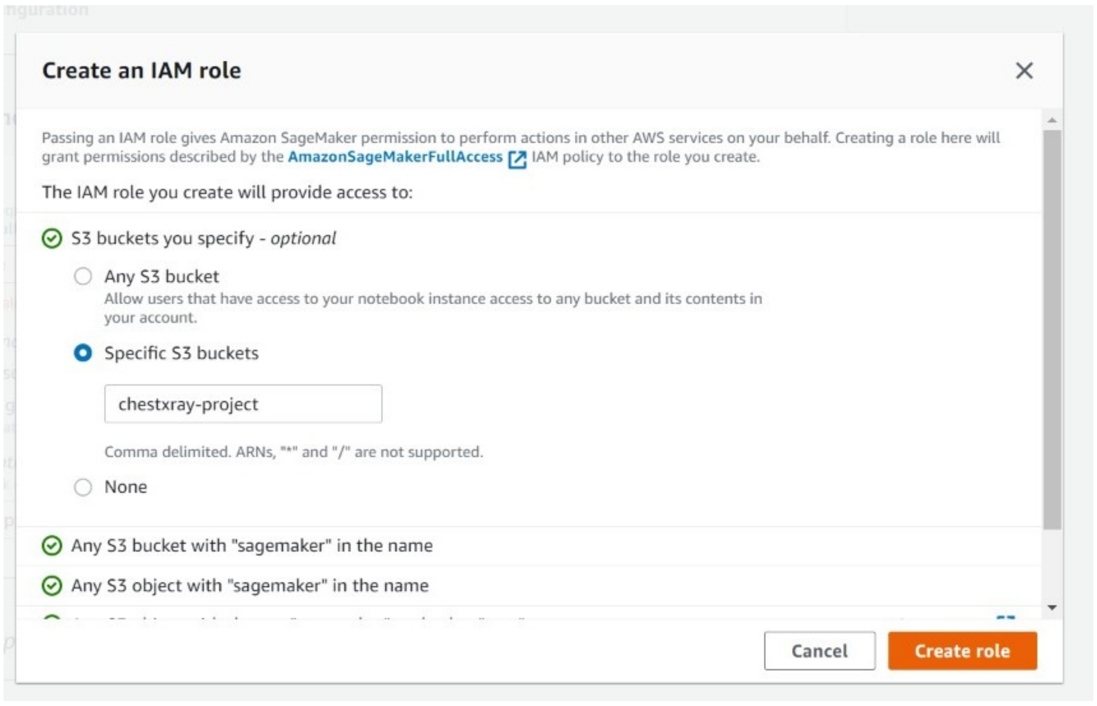
Figure 9: Account Creation

6.2.2 Create an AWS Identity and Access Management (IAM)

Choose create a new role for IAM role.

1. Select Create a new role.
2. Select a specific S3 bucket on the Create an IAM role page.
3. Select Create role next.

Using the name AmazonSageMakerExecutionRole, Amazon SageMaker creates an IAM role.



Create an IAM role

Passing an IAM role gives Amazon SageMaker permission to perform actions in other AWS services on your behalf. Creating a role here will grant permissions described by the **AmazonSageMakerFullAccess** IAM policy to the role you create.

The IAM role you create will provide access to:

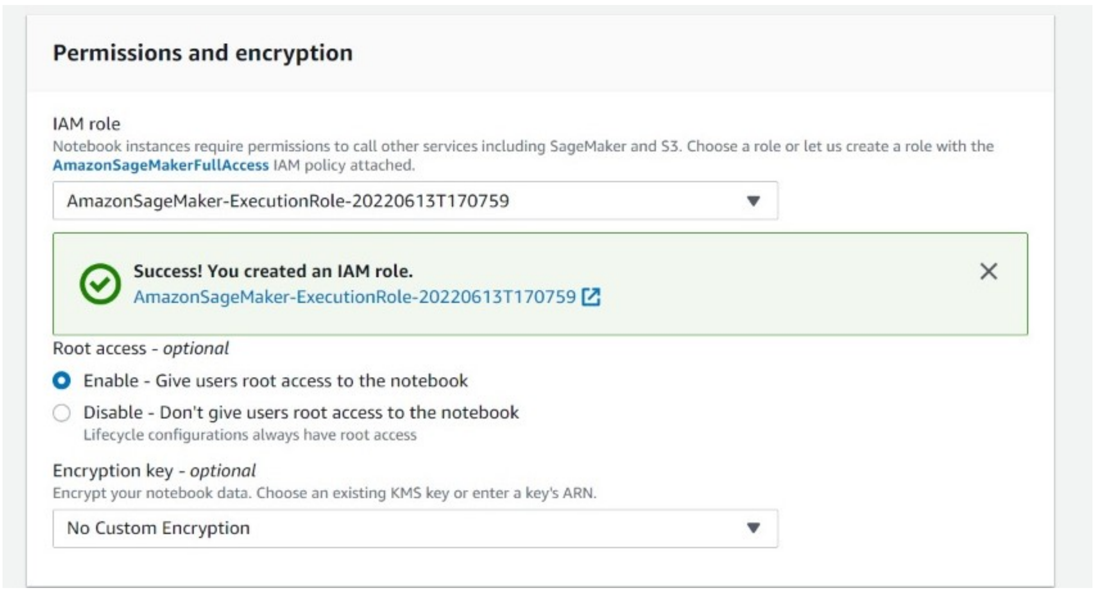
✓ S3 buckets you specify - *optional*

- ○ Any S3 bucket
  Allow users that have access to your notebook instance access to any bucket and its contents in your account.
- ● Specific S3 buckets
  chestxray-project
  Comma delimited. ARNs, "*" and "/" are not supported.
- ○ None

✓ Any S3 bucket with "sagemaker" in the name

✓ Any S3 object with "sagemaker" in the name

Figure 10: IAM Creation


Permissions and encryption

IAM role
Notebook instances require permissions to call other services including SageMaker and S3. Choose a role or let us create a role with the **AmazonSageMakerFullAccess** IAM policy attached.
AmazonSageMaker-ExecutionRole-20220613T170759

Success! You created an IAM role.
AmazonSageMaker-ExecutionRole-20220613T170759

Root access - *optional*
● Enable - Give users root access to the notebook
○ Disable - Don't give users root access to the notebook
Lifecycle configurations always have root access

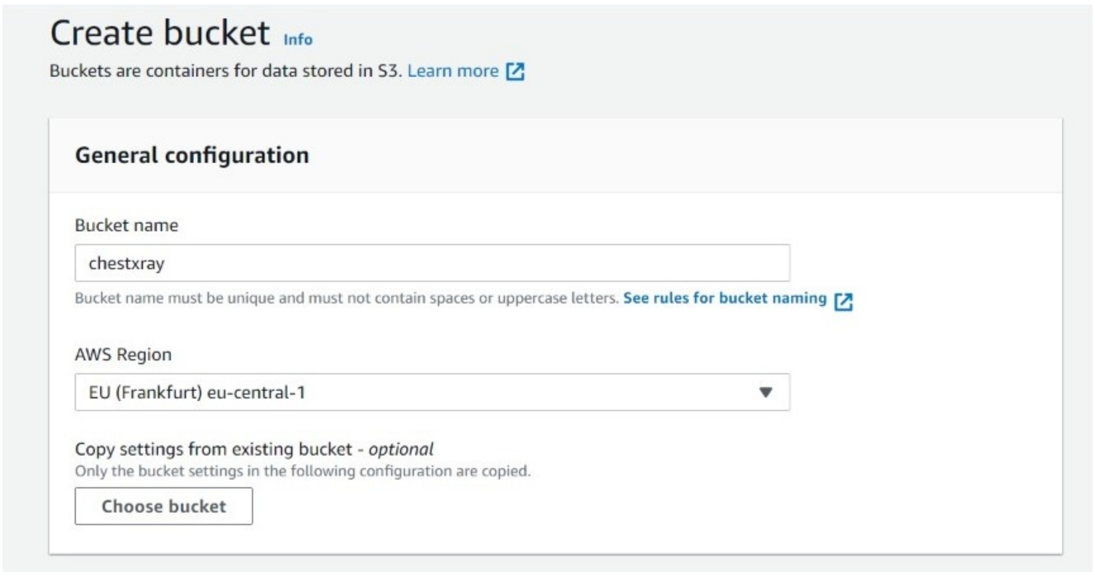
Encryption key - *optional*
Encrypt your notebook data. Choose an existing KMS key or enter a key's ARN.
No Custom Encryption

Figure 11: IAM Creation Confirmation

6.2.3 Create an Amazon S3 Bucket

Open the Amazon S3 console at https://console.aws.amazon.com/s3/ after logging into the AWS Management Console. To create a S3 bucket follow the steps:

- Select Create bucket.
- Name the bucket.
- Select the AWS Region where the bucket should be located under Region.


Create bucket Info
Buckets are containers for data stored in S3. Learn more

General configuration

Bucket name
chestxray
Bucket name must be unique and must not contain spaces or uppercase letters. **See rules for bucket naming**

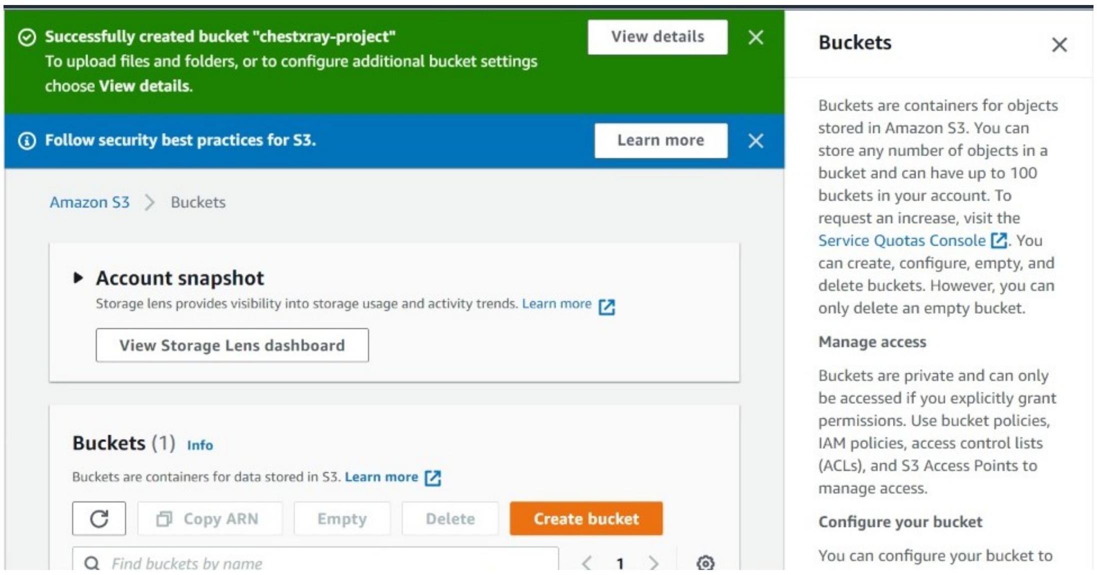
AWS Region
EU (Frankfurt) eu-central-1

Copy settings from existing bucket - *optional*
Only the bucket settings in the following configuration are copied.
Choose bucket

Figure 12: S3 Bucket Creation

- Select one of the following settings under Object Ownership to enable or disable ACLs and manage ownership of objects uploaded to your bucket [5].
- Select the Block Public Access settings that you want to apply to the bucket under Bucket settings for Block Public Access.

- Select Create bucket.


Successfully created bucket "chestxray-project"
To upload files and folders, or to configure additional bucket settings choose **View details**.

Follow security best practices for S3.

Amazon S3 > Buckets

Account snapshot
Storage lens provides visibility into storage usage and activity trends. Learn more

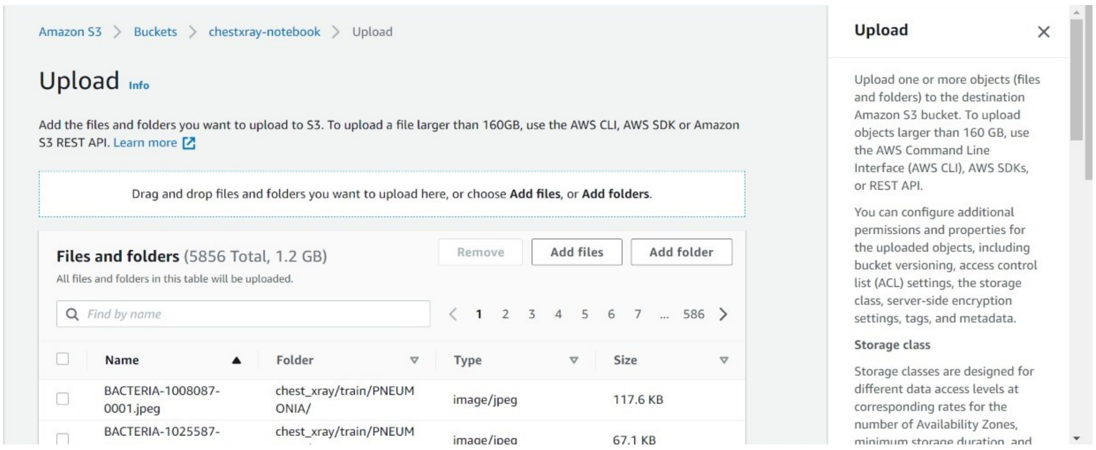
Buckets (1) Info
Buckets are containers for data stored in S3. Learn more

Buckets
Buckets are containers for objects stored in Amazon S3. You can store any number of objects in a bucket and can have up to 100 buckets in your account. To request an increase, visit the Service Quotas Console. You can create, configure, empty, and delete buckets. However, you can only delete an empty bucket.

Manage access
Buckets are private and can only be accessed if you explicitly grant permissions. Use bucket policies, IAM policies, access control lists (ACLs), and S3 Access Points to manage access.

Configure your bucket
You can configure your bucket to

Figure 13: S3 Bucket Creation Confirmation


Amazon S3 > Buckets > chestxray-notebook > Upload

Upload Info

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. Learn more

Drag and drop files and folders you want to upload here, or choose **Add files**, or **Add folders**.

Files and folders (5856 Total, 1.2 GB)
All files and folders in this table will be uploaded.

Name	Folder	Type	Size
BACTERIA-1008087-0001.jpeg	chest_xray/train/PNEUMONIA/	image/jpeg	117.6 KB
BACTERIA-1025587-	chest_xray/train/PNEUM	image/jpeg	67.1 KB

Upload
Upload one or more objects (files and folders) to the destination Amazon S3 bucket. To upload objects larger than 160 GB, use the AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API.

You can configure additional permissions and properties for the uploaded objects, including bucket versioning, access control list (ACL) settings, the storage class, server-side encryption settings, tags, and metadata.

Storage class
Storage classes are designed for different data access levels at corresponding rates for the number of Availability Zones, minimum storage duration, and

Figure 14: Store the Dataset in the S3 Bucket

- Store the dataset used for the data training and the model artifacts that a Amazon SageMaker training job outputs.

To access these buckets, Amazon SageMaker needs authorization. An IAM role, which you create in the following step when you create an Amazon SageMaker notebook instance, is how you grant permission. Any bucket with the name Sagemaker automatically grants access to this IAM role [5]. These rights are granted to it by the role's attachment of the AmazonSageMakerFullAccess policy by Amazon SageMaker.

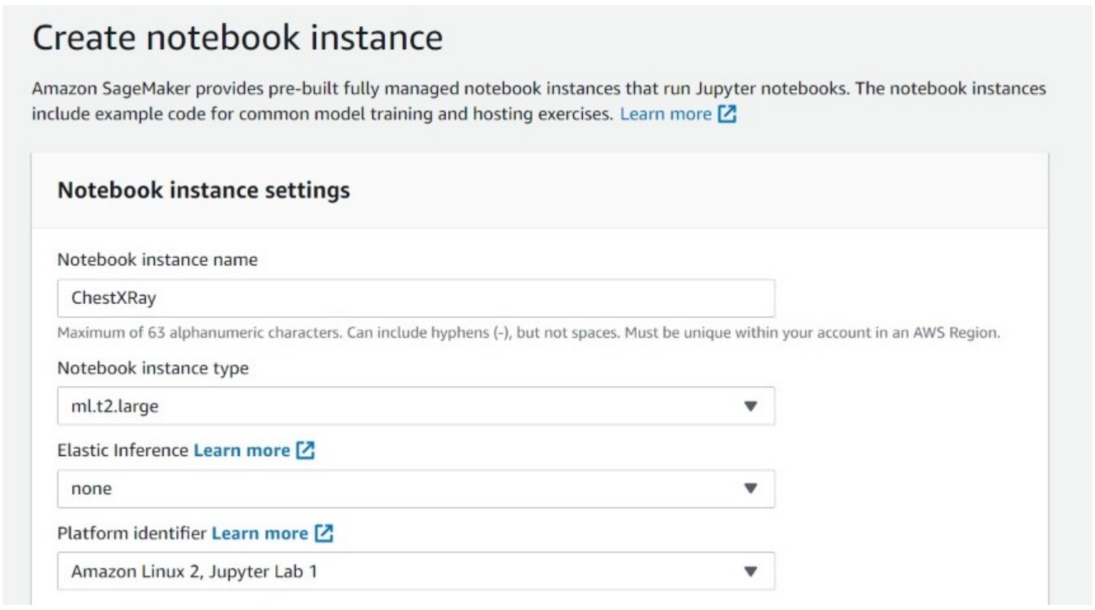
6.2.4 Create the Sagemaker Notebook Instance

1. Start Amazon Sagemaker by clicking on Find Services in the AWS Management Console.
2. Select Create notebook instance from the Notebook instances tab.
3. Give your notebook instance a name.
4. To grant the notebook instance access to your S3 bucket, choose the IAM role created in the previous step [3] [5]. Continually use the other default settings.

5. To start and create a new instance, click Create notebook instance in step. The instance's initialization may take several minutes.

6. For your Jupyter notebook to launch, click Open Jupyter.

In the newly created instance, we can now run the python code for training a model by launching the Jupyter notebook.



Create notebook instance

Amazon SageMaker provides pre-built fully managed notebook instances that run Jupyter notebooks. The notebook instances include example code for common model training and hosting exercises. Learn more

Notebook instance settings

Notebook instance name: ChestXRay

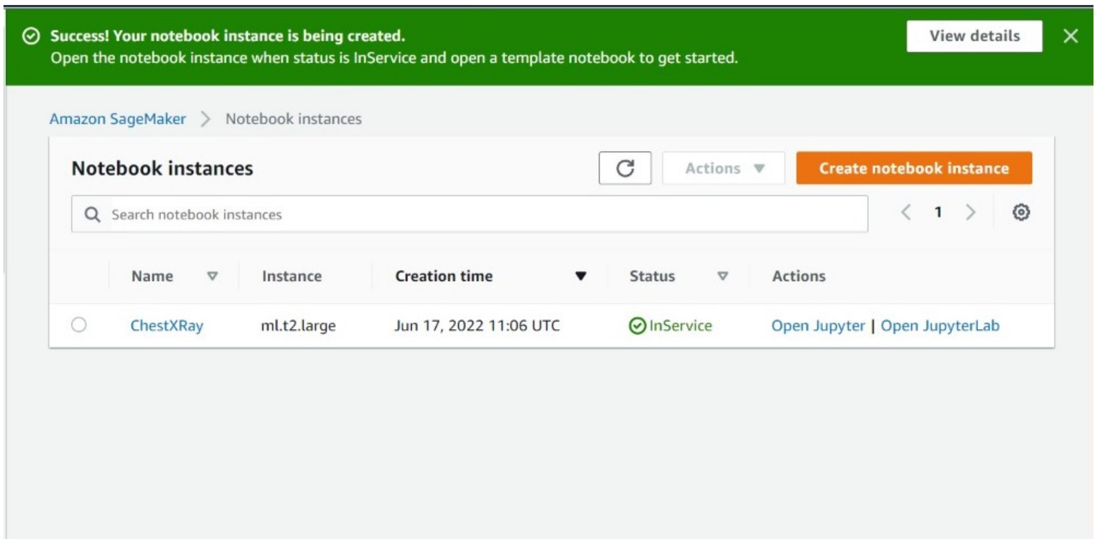
Maximum of 63 alphanumeric characters. Can include hyphens (-), but not spaces. Must be unique within your account in an AWS Region.

Notebook instance type: ml.t2.large

Elastic Inference Learn more: none

Platform identifier Learn more: Amazon Linux 2, Jupyter Lab 1

Figure 15: Notebook Instance Creation



Success! Your notebook instance is being created.
Open the notebook instance when status is InService and open a template notebook to get started.

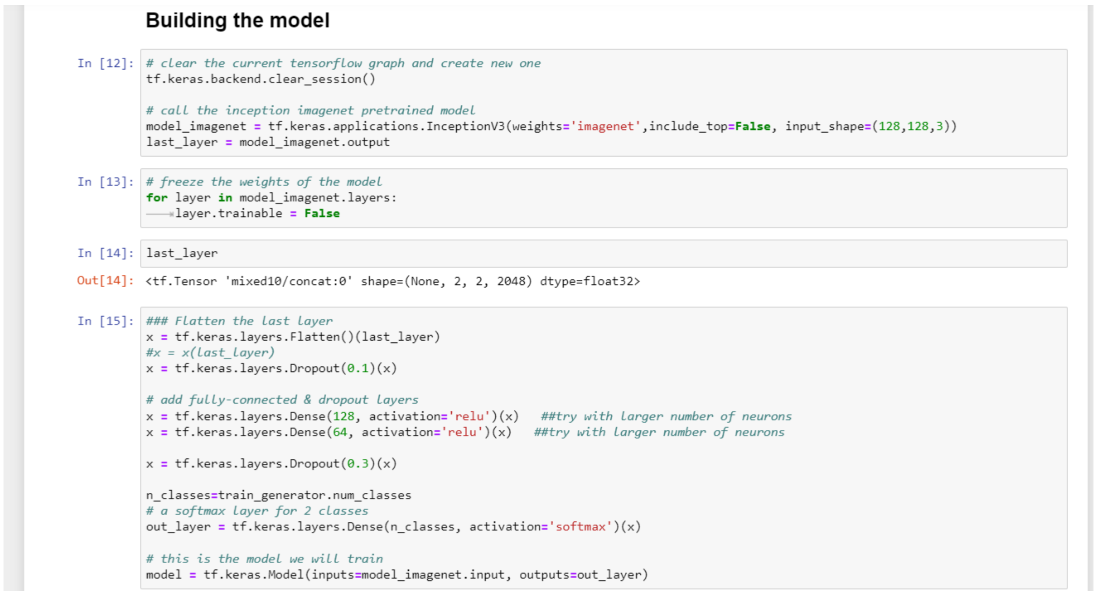
View details
Amazon SageMaker > Notebook instances
Notebook instances
Actions
Create notebook instance
Search notebook instances

Name	Instance	Creation time	Status	Actions
ChestXRay	ml.t2.large	Jun 17, 2022 11:06 UTC	InService	Open Jupyter | Open JupyterLab

Figure 16: Notebook Instance Creation Confirmation

6.2.5 Build the model

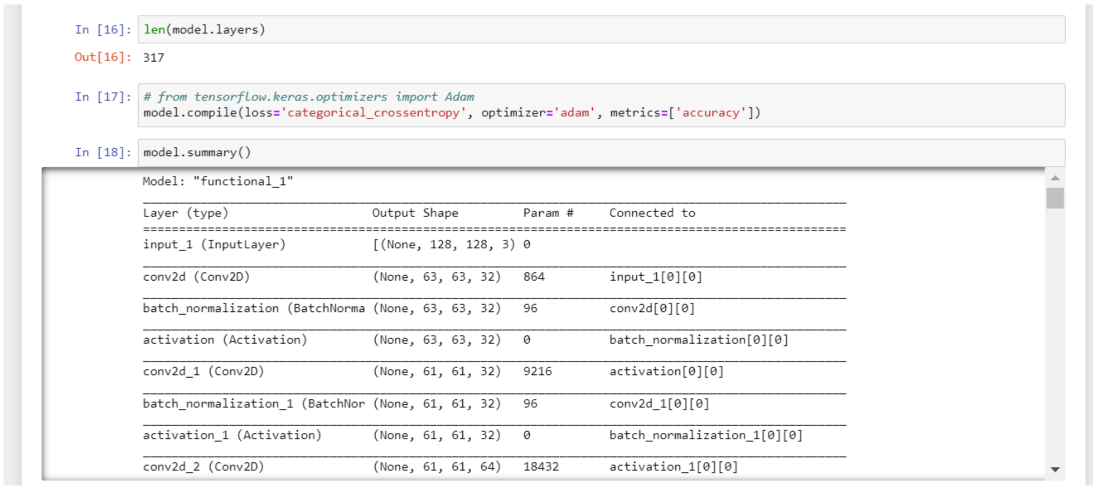
TensorFlow combines several different techniques and models, allowing users to build deep neural networks for applications such as image recognition/classification and natural language processing. Using the InceptionV3 Imagenet model, created an image classification machine learning model. We implemented the pre-trained saved model in our model because it has been trained on numerous images and has a robust neural network. Imagenet includes around 1 million images of common general things divided into 20,000 categories such as human faces, animals, toys, balloons, and so on, and it is not trained on X-rays or other medical images.



Building the model

```
In [12]: # clear the current tensorflow graph and create new one
         tf.keras.backend.clear_session()

         # call the inception imagenet pretrained model
         model_imagenet = tf.keras.applications.InceptionV3(weights='imagenet',include_top=False, input_shape=(128,128,3))
         last_layer = model_imagenet.output

In [13]: # freeze the weights of the model
         for layer in model_imagenet.layers:
             layer.trainable = False

In [14]: last_layer

Out[14]: <tf.Tensor 'mixed10/concat:0' shape=(None, 2, 2, 2048) dtype=float32>

In [15]: ### Flatten the last layer
         x = tf.keras.layers.Flatten()(last_layer)
         #x = x(last_layer)
         x = tf.keras.layers.Dropout(0.1)(x)

         # add fully-connected & dropout layers
         x = tf.keras.layers.Dense(128, activation='relu')(x)   ##try with larger number of neurons
         x = tf.keras.layers.Dense(64, activation='relu')(x)   ##try with larger number of neurons

         x = tf.keras.layers.Dropout(0.3)(x)

         n_classes=train_generator.num_classes
         # a softmax layer for 2 classes
         out_layer = tf.keras.layers.Dense(n_classes, activation='softmax')(x)

         # this is the model we will train
         model = tf.keras.Model(inputs=model_imagenet.input, outputs=out_layer)
```

Figure 17: Build the model in Jupyter Notebook

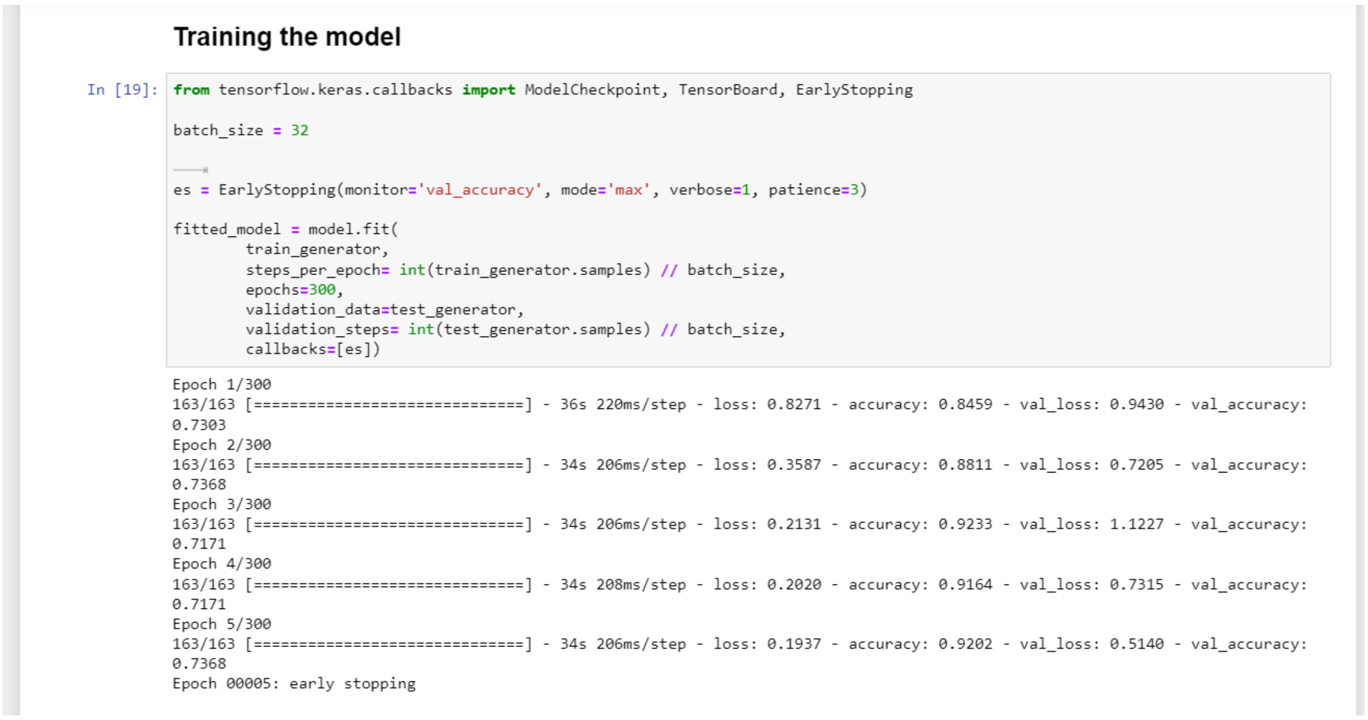


```
In [16]: len(model.layers)

Out[16]: 317

In [17]: # from tensorflow.keras.optimizers import Adam
         model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

In [18]: model.summary()

         Model: "functional_1"
         __________________________________________________________________________________________________
         Layer (type)                    Output Shape         Param #     Connected to
         ==================================================================================================
         input_1 (InputLayer)            [(None, 128, 128, 3) 0
         __________________________________________________________________________________________________
         conv2d (Conv2D)                 (None, 63, 63, 32)   864         input_1[0][0]
         __________________________________________________________________________________________________
         batch_normalization (BatchNorma (None, 63, 63, 32)   96          conv2d[0][0]
         __________________________________________________________________________________________________
         activation (Activation)         (None, 63, 63, 32)   0           batch_normalization[0][0]
         __________________________________________________________________________________________________
         conv2d_1 (Conv2D)               (None, 61, 61, 32)   9216        activation[0][0]
         __________________________________________________________________________________________________
         batch_normalization_1 (BatchNor (None, 61, 61, 32)   96          conv2d_1[0][0]
         __________________________________________________________________________________________________
         activation_1 (Activation)       (None, 61, 61, 32)   0           batch_normalization_1[0][0]
         __________________________________________________________________________________________________
         conv2d_2 (Conv2D)               (None, 61, 61, 64)   18432       activation_1[0][0]
```

Figure 18: Build the model in Jupyter Notebook

6.2.6 Train the Model

Adam was chosen as our optimizer since it uses the gradient descent approach. For our compute instance, we've set the batch size to 32 and the epochs to 3, which indicates how many training steps our model will go through. We used a Jupyter notebook in our SageMaker notebook instance to train and test our model.



Training the model

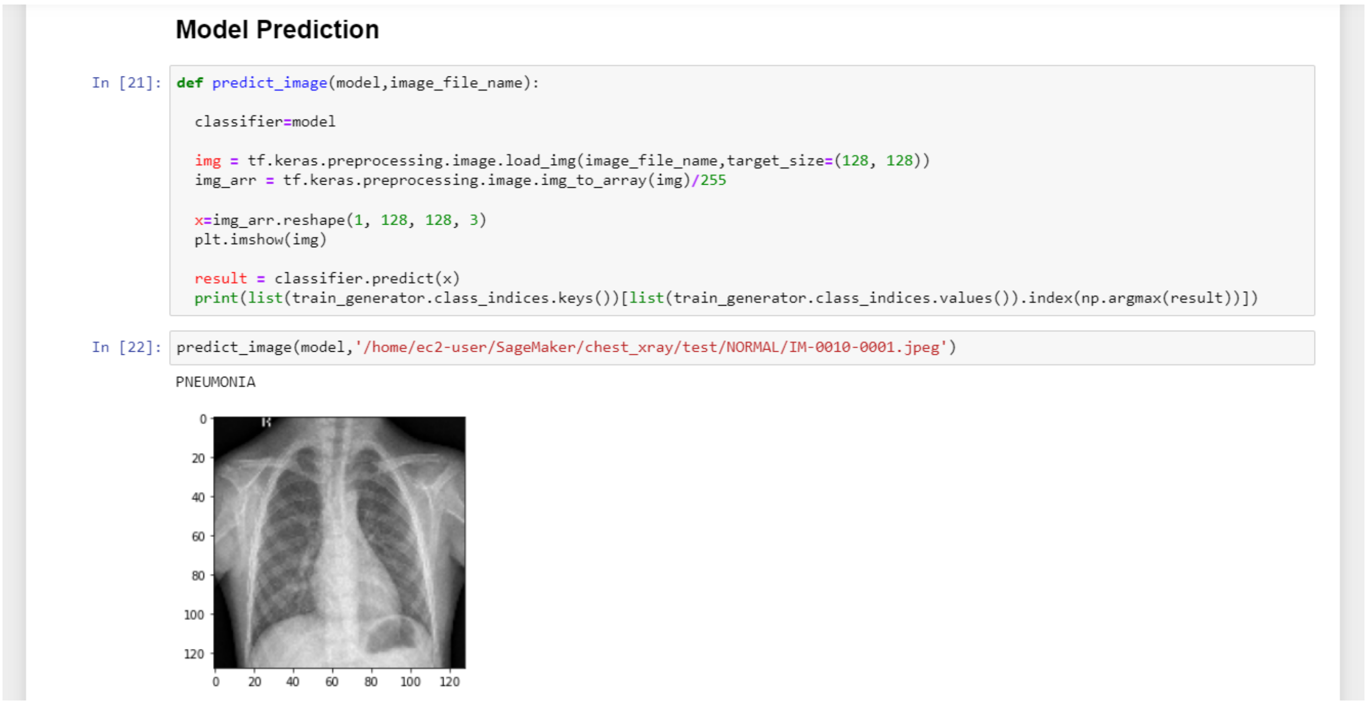
```
In [19]: from tensorflow.keras.callbacks import ModelCheckpoint, TensorBoard, EarlyStopping

         batch_size = 32


         es = EarlyStopping(monitor='val_accuracy', mode='max', verbose=1, patience=3)

         fitted_model = model.fit(
                 train_generator,
                 steps_per_epoch= int(train_generator.samples) // batch_size,
                 epochs=300,
                 validation_data=test_generator,
                 validation_steps= int(test_generator.samples) // batch_size,
                 callbacks=[es])
```

```
Epoch 1/300
163/163 [==============================] - 36s 220ms/step - loss: 0.8271 - accuracy: 0.8459 - val_loss: 0.9430 - val_accuracy:
0.7303
Epoch 2/300
163/163 [==============================] - 34s 206ms/step - loss: 0.3587 - accuracy: 0.8811 - val_loss: 0.7205 - val_accuracy:
0.7368
Epoch 3/300
163/163 [==============================] - 34s 206ms/step - loss: 0.2131 - accuracy: 0.9233 - val_loss: 1.1227 - val_accuracy:
0.7171
Epoch 4/300
163/163 [==============================] - 34s 208ms/step - loss: 0.2020 - accuracy: 0.9164 - val_loss: 0.7315 - val_accuracy:
0.7171
Epoch 5/300
163/163 [==============================] - 34s 206ms/step - loss: 0.1937 - accuracy: 0.9202 - val_loss: 0.5140 - val_accuracy:
0.7368
Epoch 00005: early stopping
```

Figure 19: Train the Model in Jupyter Notebook

6.2.7 Predict and Evaluate the Model

This step enables the assessment of the effectiveness and accuracy of the machine learning models. Users can run the code below in their Jupyter notebook. We provided test photos to predict and utilize for evaluation.



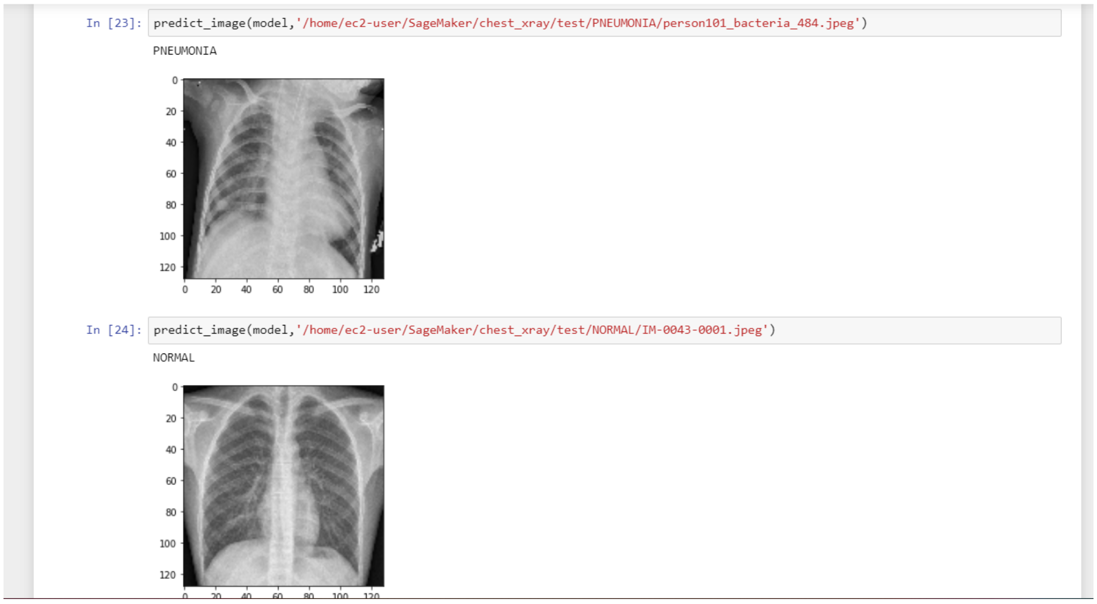
Model Prediction

```
In [21]: def predict_image(model,image_file_name):

           classifier=model

           img = tf.keras.preprocessing.image.load_img(image_file_name,target_size=(128, 128))
           img_arr = tf.keras.preprocessing.image.img_to_array(img)/255

           x=img_arr.reshape(1, 128, 128, 3)
           plt.imshow(img)

           result = classifier.predict(x)
           print(list(train_generator.class_indices.keys())[list(train_generator.class_indices.values()).index(np.argmax(result))])
```

```
In [22]: predict_image(model,'/home/ec2-user/SageMaker/chest_xray/test/NORMAL/IM-0010-0001.jpeg')
```

```
PNEUMONIA
```

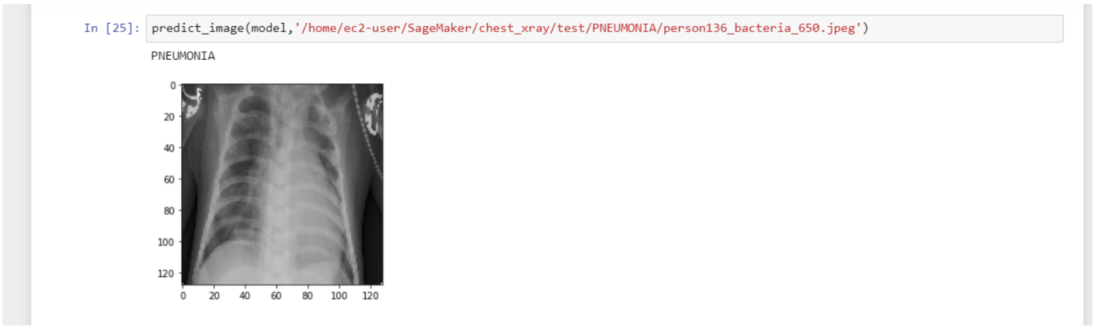
Figure 20: Predict the Model



```
In [23]: predict_image(model,'/home/ec2-user/SageMaker/chest_xray/test/PNEUMONIA/person101_bacteria_484.jpeg')
PNEUMONIA
```

```
In [24]: predict_image(model,'/home/ec2-user/SageMaker/chest_xray/test/NORMAL/IM-0043-0001.jpeg')
NORMAL
```

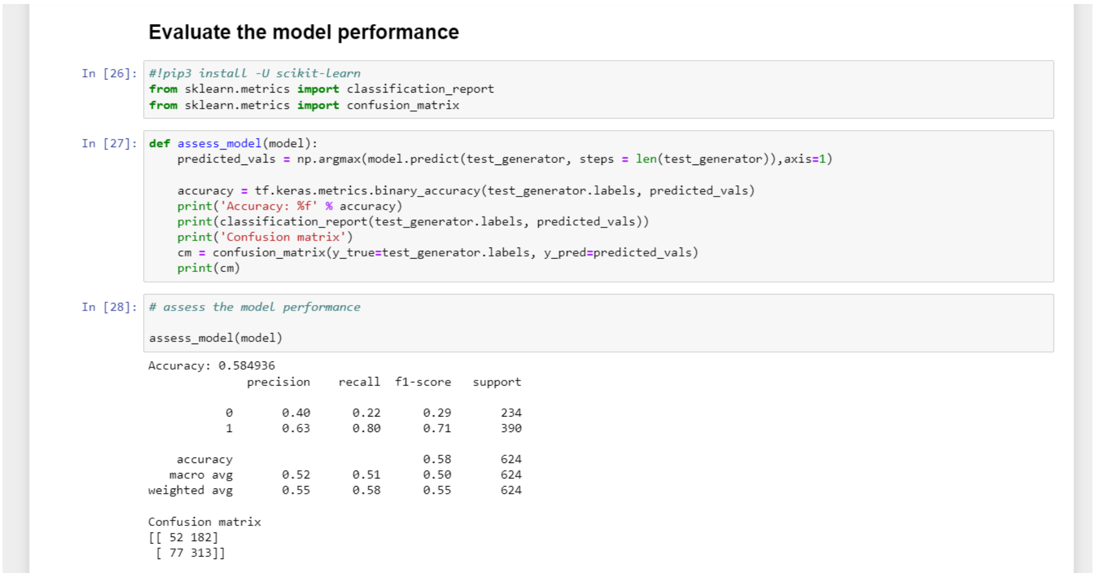
Figure 21: Predict the Model



```
In [25]: predict_image(model,'/home/ec2-user/SageMaker/chest_xray/test/PNEUMONIA/person136_bacteria_650.jpeg')
PNEUMONIA
```

Figure 22: Predict the Model

This code compares the actual and predicted values in a table known as the confusion matrix. Based on the prediction, the user can determine whether or not a predicted chest X-ray shows pneumonia.



Evaluate the model performance

```
In [26]: #!pip3 install -U scikit-learn
         from sklearn.metrics import classification_report
         from sklearn.metrics import confusion_matrix

In [27]: def assess_model(model):
             predicted_vals = np.argmax(model.predict(test_generator, steps = len(test_generator)),axis=1)

             accuracy = tf.keras.metrics.binary_accuracy(test_generator.labels, predicted_vals)
             print('Accuracy: %f' % accuracy)
             print(classification_report(test_generator.labels, predicted_vals))
             print('Confusion matrix')
             cm = confusion_matrix(y_true=test_generator.labels, y_pred=predicted_vals)
             print(cm)

In [28]: # assess the model performance

         assess_model(model)

Accuracy: 0.584936
              precision    recall  f1-score   support

           0       0.40      0.22      0.29       234
           1       0.63      0.80      0.71       390

    accuracy                           0.58       624
   macro avg       0.52      0.51      0.50       624
weighted avg       0.55      0.58      0.55       624

Confusion matrix
[[ 52 182]
 [ 77 313]]
```

Figure 23: Evaluate the Model

6.3 Azure Implementation

This section demonstrates how to construct the entire AzureML workflow using Python AzureML tools, from creating datasets to deriving final models using a web service [3]. Azure has created a number of python packages that allow users to access and authenticate AzureML services.

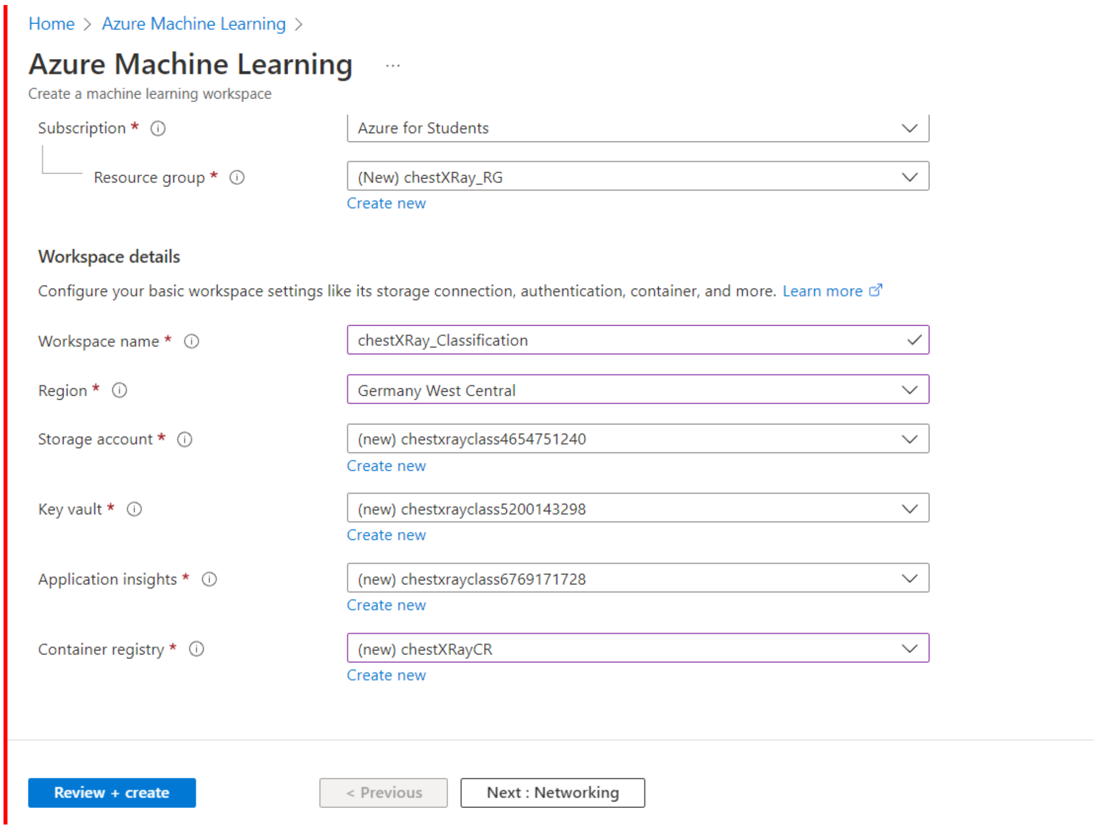
6.3.1 Azure Authentication

In order to use Azure's services, we must authenticate with it. Azure provides a variety of methods for logging into your account and authenticating yourself. We can either use Azure Service Principal authentication or Azure Interactive Login authentication, which is our choice.

6.3.2 Create Workspace

A workspace needs a variety of resources, such as keyvaults, or storage. We must pass a few arguments containing these values to the method when creating the workspace.

1. Name the workspace.
2. The Subscription ID can be found in our subscription ID on the Azure portal.
3. We list the name of the group of resources we are using.
4. Details of the authentication that we completed in a previous step.
5. Put the location on the cloud that we'll be using.


Home > Azure Machine Learning >
Azure Machine Learning
Create a machine learning workspace

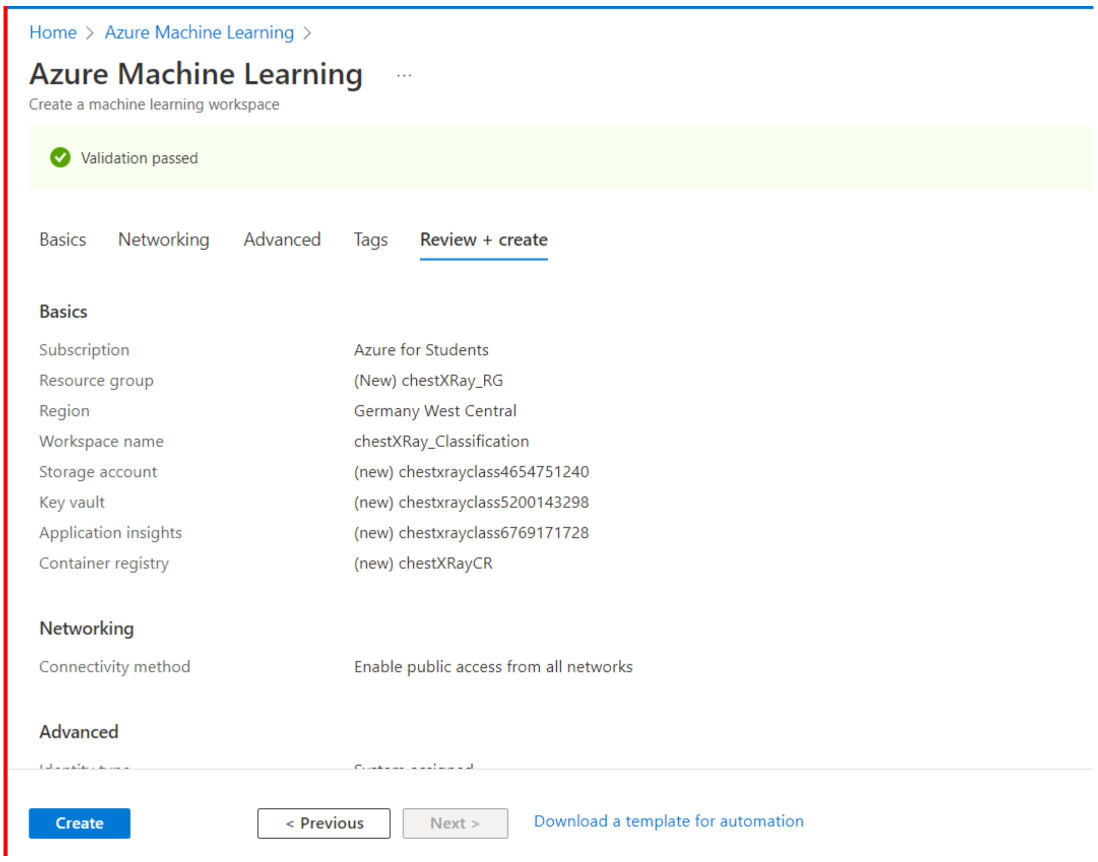
Subscription *	Azure for Students
Resource group *	(New) chestXRay_RG

Workspace details

Configure your basic workspace settings like its storage connection, authentication, container, and more. Learn more

Workspace name *	chestXRay_Classification
Region *	Germany West Central
Storage account *	(new) chestxrayclass4654751240
Key vault *	(new) chestxrayclass5200143298
Application insights *	(new) chestxrayclass6769171728
Container registry *	(new) chestXRayCR

Figure 24: Workspace Creation



Basics

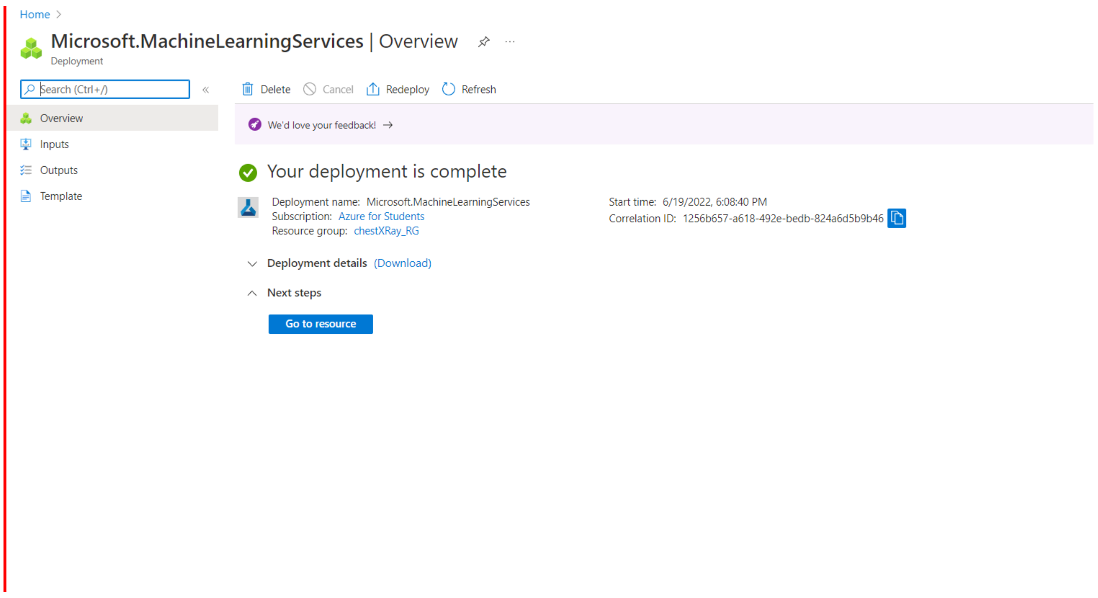
Subscription	Azure for Students
Resource group	(New) chestXRay_RG
Region	Germany West Central
Workspace name	chestXRay_Classification
Storage account	(new) chestxrayclass4654751240
Key vault	(new) chestxrayclass5200143298
Application insights	(new) chestxrayclass6769171728
Container registry	(new) chestXRayCR

Networking

Connectivity method	Enable public access from all networks

Advanced

Figure 25: Workspace Creation

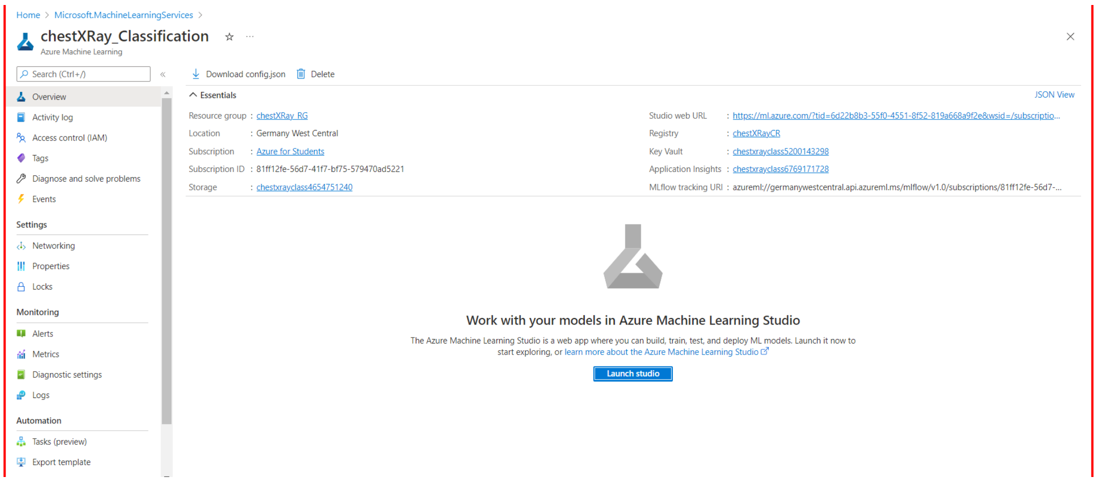


Your deployment is complete

Deployment name: Microsoft.MachineLearningServices
Subscription: Azure for Students
Resource group: chestXRay_RG

Start time: 6/19/2022, 6:08:40 PM
Correlation ID: 1256b657-a618-492e-bedb-824a6d5b9b46

Figure 26: Workspace Creation


Home > Microsoft.MachineLearningServices >

chestXRay_Classification

Azure Machine Learning

Download config.json | Delete

Essentials

Resource group	chestXRay_RG
Location	Germany West Central
Subscription	Azure for Students
Subscription ID	81ff12fe-56d7-41f7-bf75-579470ad5221
Storage	chestxrayclass4654751240
Studio web URL	https://ml.azure.com/?tid=6d22b8b3-55f0-4551-8f52-819a668a9f2e&wsid=/subscriptio...
Registry	chestXRayCR
Key Vault	chestxrayclass5200143298
Application Insights	chestxrayclass6769171728
MLflow tracking URI	azureml://germanywestcentral.api.azureml.ms/mlflow/v1.0/subscriptions/81ff12fe-56d7-...

Work with your models in Azure Machine Learning Studio

The Azure Machine Learning Studio is a web app where you can build, train, test, and deploy ML models. Launch it now to start exploring, or learn more about the Azure Machine Learning Studio

Figure 27: Workspace Creation

6.3.3 Create Compute Instance

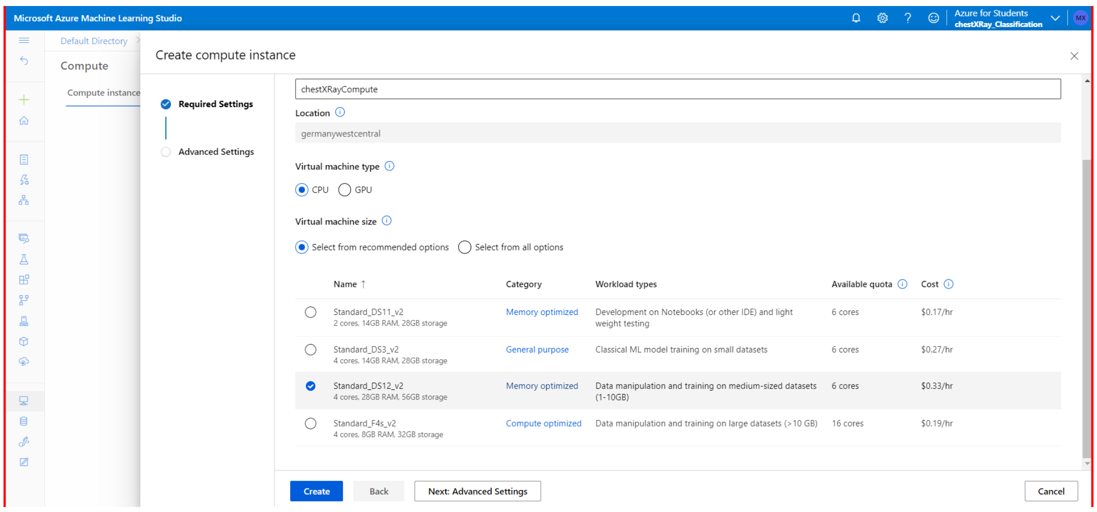
An Azure Machine Learning compute instance is a managed cloud-based workstation for data scientists.

1. Select the New link next to the Compute name dropdown in the Create New Experiment dialog.

2. Enter the Compute name in the text box of the Create New Compute dialog.

3. From the drop-down menu for Compute size, select Standard DS12_v2. The compute types used by Model Builder are CPU-optimized.

4. Choose Create. The provisioning of the compute resources could take a few minutes.


Microsoft Azure Machine Learning Studio

Create compute instance

chestXRayCompute

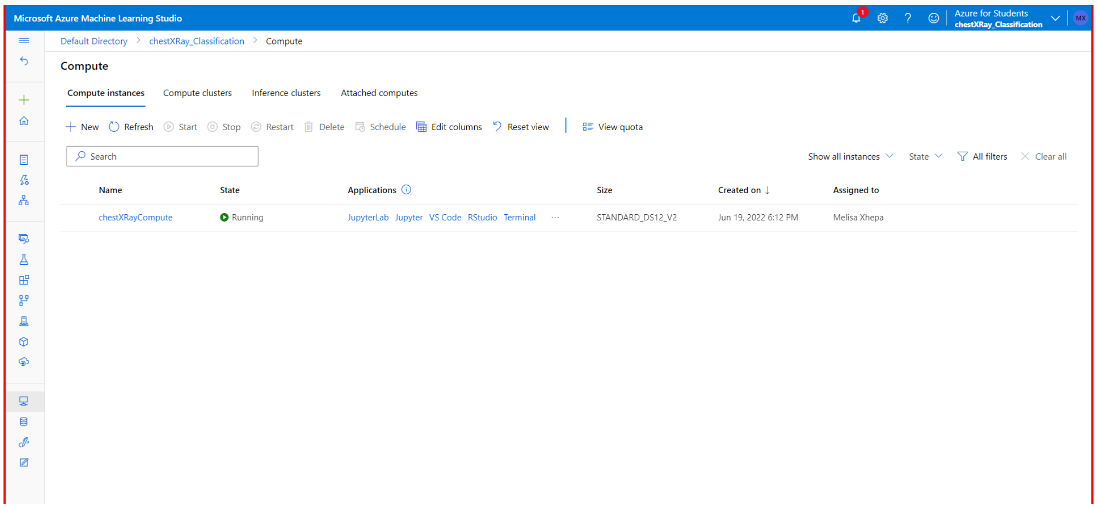
Location: germanywestcentral

Virtual machine type: [✓ CPU] GPU

Virtual machine size: [✓ Select from recommended options] Select from all options

	Name ↑	Category	Workload types	Available quota	Cost
	Standard_DS11_v2 2 cores, 14GB RAM, 28GB storage	Memory optimized	Development on Notebooks (or other IDE) and light weight testing	6 cores	$0.17/hr
	Standard_DS3_v2 4 cores, 14GB RAM, 28GB storage	General purpose	Classical ML model training on small datasets	6 cores	$0.27/hr
✓	Standard_DS12_v2 4 cores, 28GB RAM, 56GB storage	Memory optimized	Data manipulation and training on medium-sized datasets (1-10GB)	6 cores	$0.33/hr
	Standard_F4s_v2 4 cores, 8GB RAM, 32GB storage	Compute optimized	Data manipulation and training on large datasets (>10 GB)	16 cores	$0.19/hr

Figure 28: Compute Instance Creation


Microsoft Azure Machine Learning Studio

Azure for Students
chestXRay_Classification

Default Directory > chestXRay_Classification > Compute

Compute

Compute instances | Compute clusters | Inference clusters | Attached computes

Name	State	Applications	Size	Created on	Assigned to
chestXRayCompute	Running	JupyterLab Jupyter VS Code RStudio Terminal ···	STANDARD_DS12_V2	Jun 19, 2022 6:12 PM	Melisa Xhepa

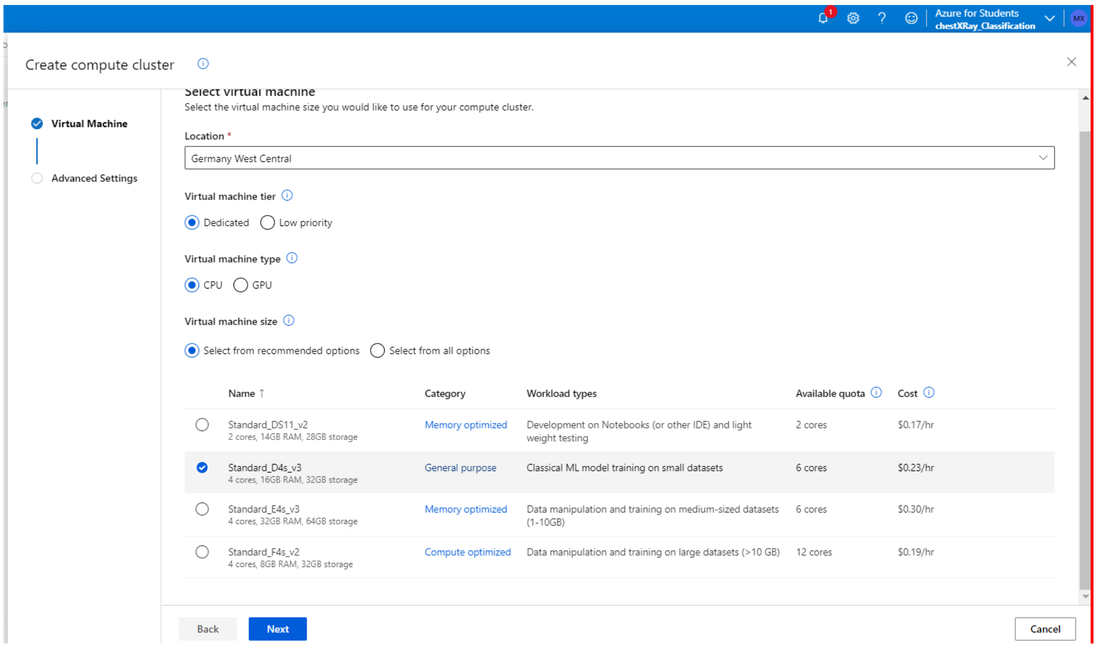
Figure 29: Compute Instance Creation Confirmation

5. Choose your newly created workspace from the Compute name dropdown in the Create New Experiment dialog after the provisioning process is finished.

6. To load the data, select the Next step button.

6.3.4 Create Compute Cluster

Azure provides different specification CPU and GPU VM clusters that can be built to meet our computing needs [9]. Regarding its compute class, each virtual machine is given a name, and its pricing varies. From Azure, we can get complete information on current pricing and compute VM availability.


Azure for Students
chestXRay_Classification

Create compute cluster

Select virtual machine

Select the virtual machine size you would like to use for your compute cluster.

Location *: Germany West Central

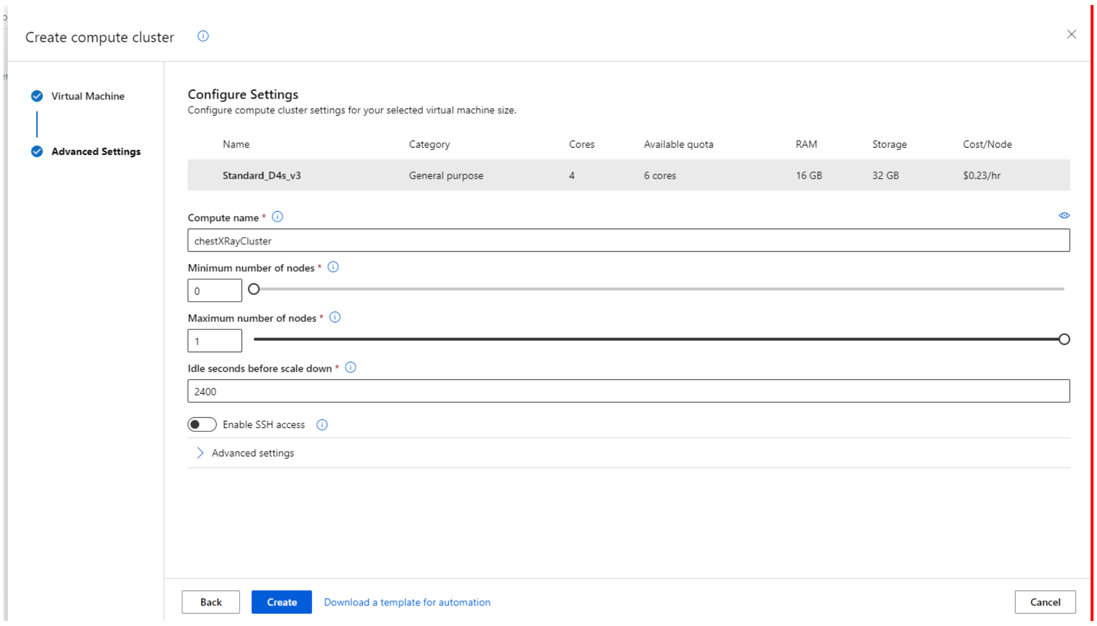
Virtual machine tier: [✓ Dedicated] Low priority

Virtual machine type: [✓ CPU] GPU

Virtual machine size: [✓ Select from recommended options] Select from all options

	Name	Category	Workload types	Available quota	Cost
	Standard_DS11_v2 2 cores, 14GB RAM, 28GB storage	Memory optimized	Development on Notebooks (or other IDE) and light weight testing	2 cores	$0.17/hr
✓	Standard_D4s_v3 4 cores, 16GB RAM, 32GB storage	General purpose	Classical ML model training on small datasets	6 cores	$0.23/hr
	Standard_E4s_v3 4 cores, 32GB RAM, 64GB storage	Memory optimized	Data manipulation and training on medium-sized datasets (1-10GB)	6 cores	$0.30/hr
	Standard_F4s_v2 4 cores, 8GB RAM, 32GB storage	Compute optimized	Data manipulation and training on large datasets (>10 GB)	12 cores	$0.19/hr

Figure 30: Compute Cluster Creation


Create compute cluster

Configure Settings

Configure compute cluster settings for your selected virtual machine size.

Name	Category	Cores	Available quota	RAM	Storage	Cost/Node
Standard_D4s_v3	General purpose	4	6 cores	16 GB	32 GB	$0.23/hr

Compute name * : chestXRayCluster

Minimum number of nodes * : 0

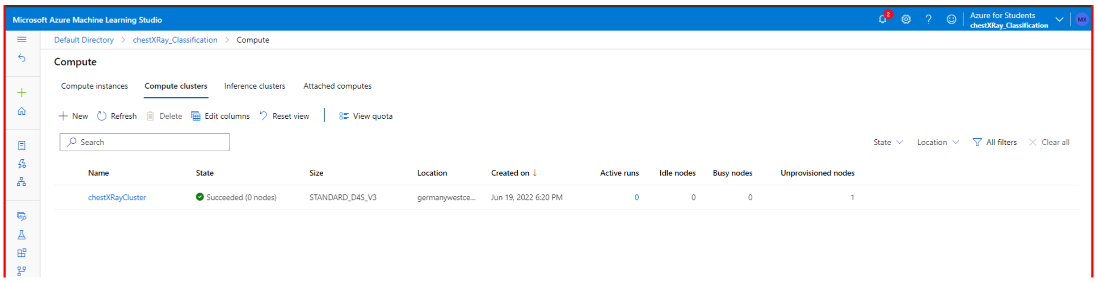
Maximum number of nodes * : 1

Idle seconds before scale down * : 2400

Enable SSH access

Figure 31: Compute Cluster Creation

Due to the fact that we will be building a straightforward image classification model, we build a CPU cluster for this project. If a cluster has already been created, we can retrieve it using the cluster's name and use it to train the model.

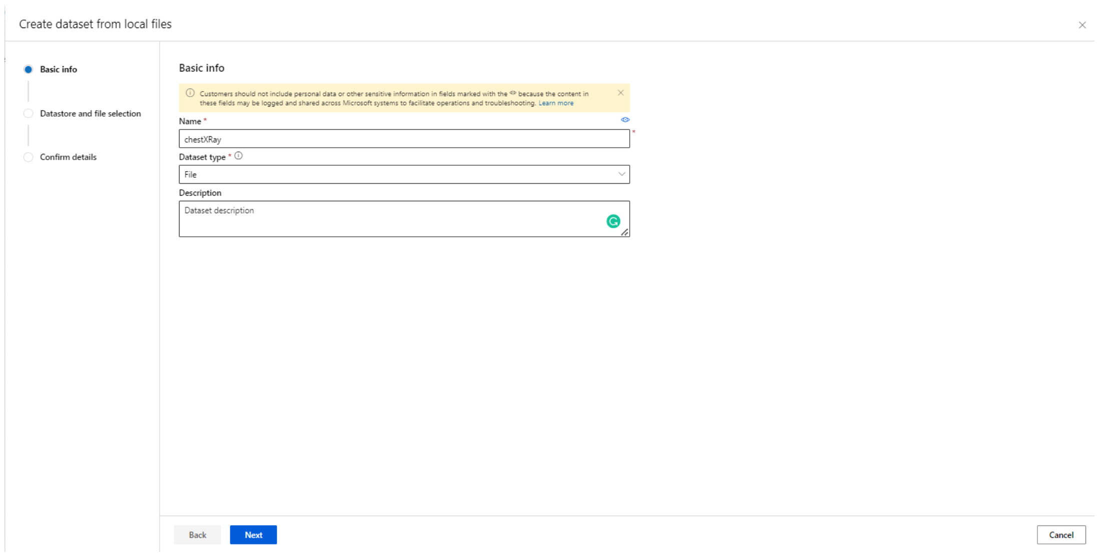


Name	State	Size	Location	Created on ↓	Active runs	Idle nodes	Busy nodes	Unprovisioned nodes
chestXRayCluster	Succeeded (0 nodes)	STANDARD_D4S_V3	germanywestce...	Jun 19, 2022 6:20 PM	0	0	0	1

Figure 32: Compute Cluster Creation Confirmation

6.3.5 Create Dataset

Depending on the type of data and the location of the data, we can use datasets in AzureML in a variety of ways when training models. Additionally, Azure provides a dataset feature in AzureML that makes it simple and low-latency to fetch and provide data [7] to training scripts. Therefore, for this, we will store our data in an Azure blob [9] and use it to train a model. Data must first be uploaded to Azure Blob Storage. Following the upload of your image files to blob storage, you should create and register your dataset. To create a dataset and register it so we can access it later in the training script, we first register a blob datastore.


Create dataset from local files

Basic info

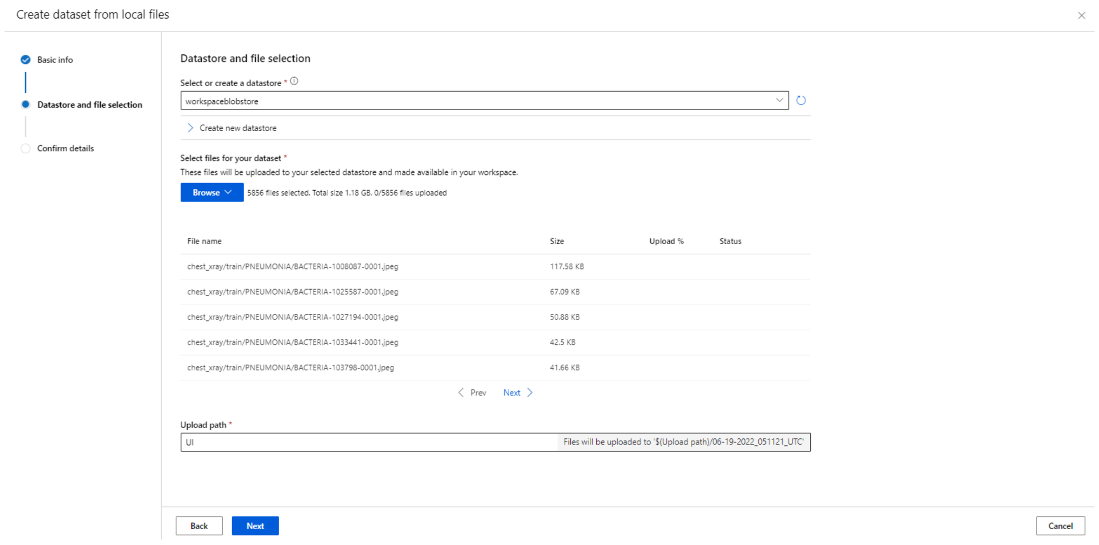
Customers should not include personal data or other sensitive information in fields marked with the because the content in these fields may be logged and shared across Microsoft systems to facilitate operations and troubleshooting. Learn more

Name *
chestXRay

Dataset type *
File

Description
Dataset description

Figure 33: Dataset Creation


Create dataset from local files

Datastore and file selection

Select or create a datastore *
workspaceblobstore

Create new datastore

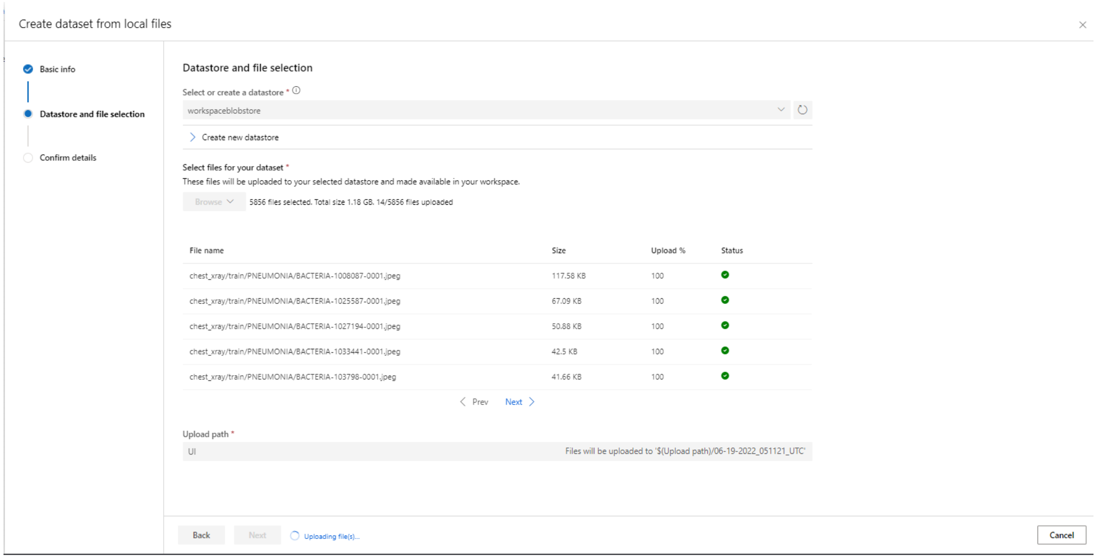
Select files for your dataset *
These files will be uploaded to your selected datastore and made available in your workspace.

Browse 5856 files selected. Total size 1.18 GB. 0/5856 files uploaded

File name	Size	Upload %	Status
chest_xray/train/PNEUMONIA/BACTERIA-1008087-0001.jpeg	117.58 KB		
chest_xray/train/PNEUMONIA/BACTERIA-1025587-0001.jpeg	67.09 KB		
chest_xray/train/PNEUMONIA/BACTERIA-1027194-0001.jpeg	50.88 KB		
chest_xray/train/PNEUMONIA/BACTERIA-1033441-0001.jpeg	42.5 KB		
chest_xray/train/PNEUMONIA/BACTERIA-103798-0001.jpeg	41.66 KB		

Upload path *
UI
Files will be uploaded to '$(Upload path)/06-19-2022_051121_UTC'

Figure 34: Dataset Creation


Create dataset from local files

Basic info
Datastore and file selection
Confirm details

Datastore and file selection

Select or create a datastore *
workspaceblobstore
Create new datastore

Select files for your dataset *
These files will be uploaded to your selected datastore and made available in your workspace.
Browse 5856 files selected. Total size 1.18 GB. 14/5856 files uploaded

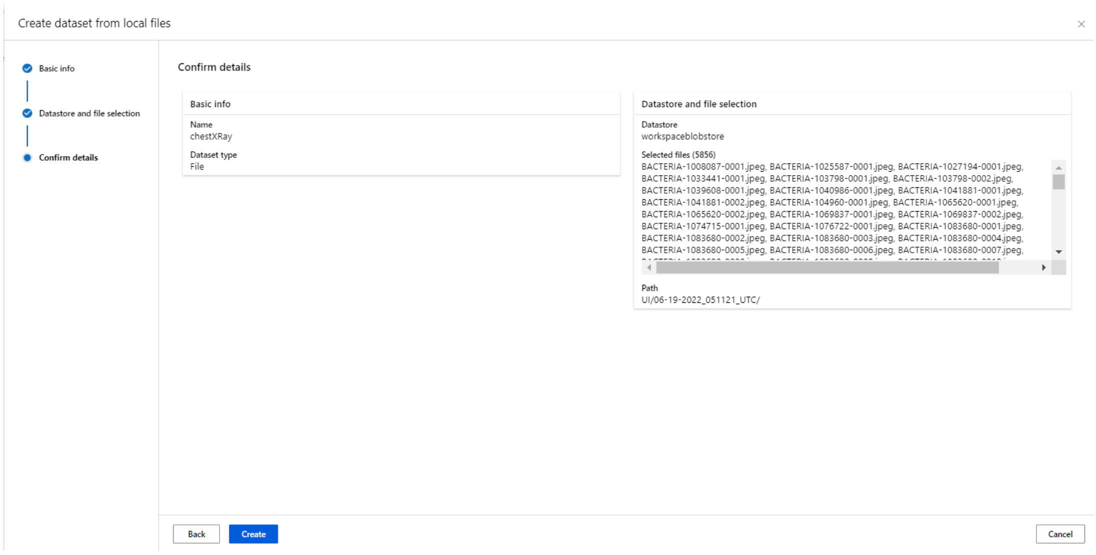
File name	Size	Upload %	Status
chest_xray/train/PNEUMONIA/BACTERIA-1008087-0001.jpeg	117.58 KB	100	✓
chest_xray/train/PNEUMONIA/BACTERIA-1025587-0001.jpeg	67.09 KB	100	✓
chest_xray/train/PNEUMONIA/BACTERIA-1027194-0001.jpeg	50.88 KB	100	✓
chest_xray/train/PNEUMONIA/BACTERIA-1033441-0001.jpeg	42.5 KB	100	✓
chest_xray/train/PNEUMONIA/BACTERIA-103798-0001.jpeg	41.66 KB	100	✓

Prev Next

Upload path *
UI Files will be uploaded to '$(Upload path)/06-19-2022_051121_UTC'

Back Next Uploading file(s)... Cancel

Figure 35: Dataset Creation


Create dataset from local files

Basic info
Datastore and file selection
Confirm details

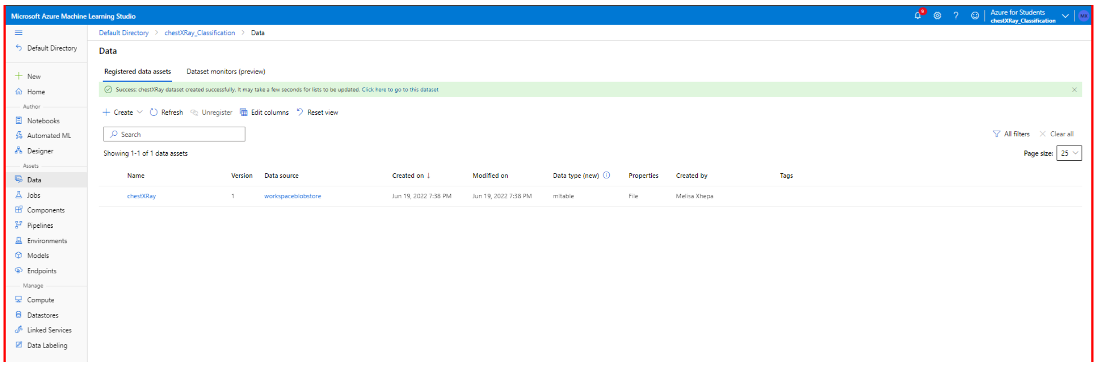
Confirm details

Basic info
Name
chestXRay
Dataset type
File

Datastore and file selection
Datastore
workspaceblobstore
Selected files (5856)
BACTERIA-1008087-0001.jpeg, BACTERIA-1025587-0001.jpeg, BACTERIA-1027194-0001.jpeg,
BACTERIA-1033441-0001.jpeg, BACTERIA-103798-0001.jpeg, BACTERIA-103798-0002.jpeg,
BACTERIA-1039608-0001.jpeg, BACTERIA-1040986-0001.jpeg, BACTERIA-1041881-0001.jpeg,
BACTERIA-1041881-0002.jpeg, BACTERIA-104960-0001.jpeg, BACTERIA-1065620-0001.jpeg,
BACTERIA-1065620-0002.jpeg, BACTERIA-1069837-0001.jpeg, BACTERIA-1069837-0002.jpeg,
BACTERIA-1074715-0001.jpeg, BACTERIA-1076722-0001.jpeg, BACTERIA-1083680-0001.jpeg,
BACTERIA-1083680-0002.jpeg, BACTERIA-1083680-0003.jpeg, BACTERIA-1083680-0004.jpeg,
BACTERIA-1083680-0005.jpeg, BACTERIA-1083680-0006.jpeg, BACTERIA-1083680-0007.jpeg,
Path
UI/06-19-2022_051121_UTC/

Back Create Cancel

Figure 36: Dataset Creation


Microsoft Azure Machine Learning Studio Azure for Students chestXRay_Classification

Default Directory > chestXRay_Classification > Data

Default Directory
New
Home
Author
Notebooks
Automated ML
Designer
Assets
Data
Jobs
Components
Pipelines
Environments
Models
Endpoints
Manage
Compute
Datastores
Linked Services
Data Labeling

Data

Registered data assets Dataset monitors (preview)

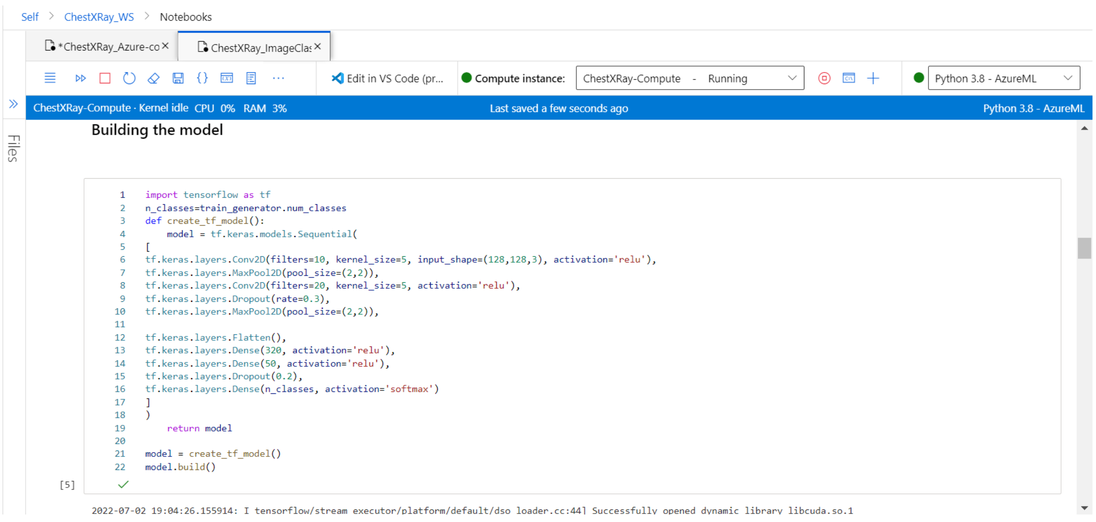
Success: chestXRay dataset created successfully. It may take a few seconds for lists to be updated. Click here to go to this dataset

Create Refresh Unregister Edit columns Reset view
Search All filters Clear all
Showing 1-1 of 1 data assets Page size: 25

Name	Version	Data source	Created on ↓	Modified on	Data type (new)	Properties	Created by	Tags
chestXRay	1	workspaceblobstore	Jun 19, 2022 7:38 PM	Jun 19, 2022 7:38 PM	mltable	File	Melisa Xhepa	

Figure 37: Dataset Creation Confirmation

6.3.6 Build Model



Compute instance: ChestXRay-Compute - Running

Python 3.8 - AzureML

ChestXRay-Compute · Kernel idle CPU 0% RAM 3% Last saved a few seconds ago Python 3.8 - AzureML

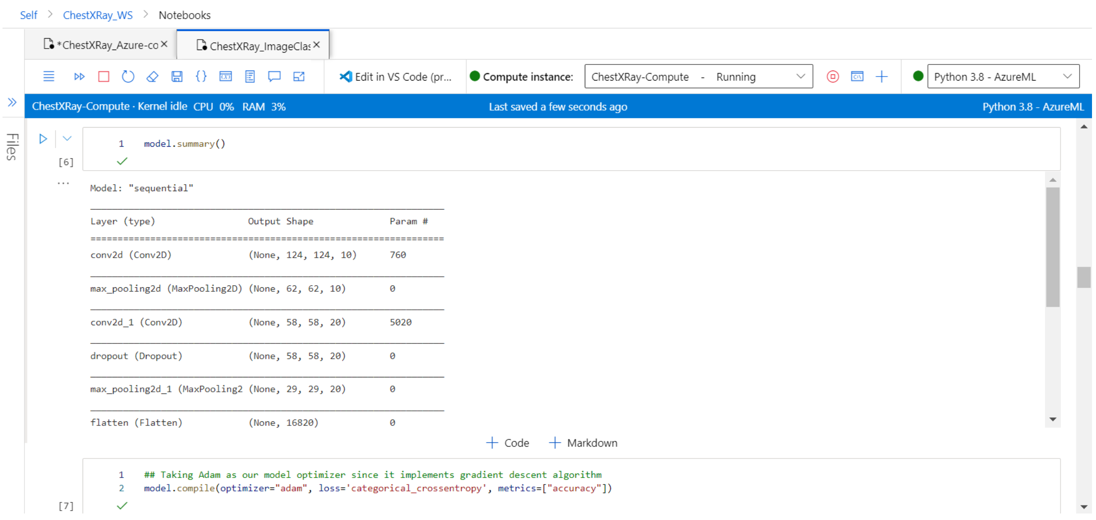
Building the model

```
import tensorflow as tf
n_classes=train_generator.num_classes
def create_tf_model():
    model = tf.keras.models.Sequential(
[
tf.keras.layers.Conv2D(filters=10, kernel_size=5, input_shape=(128,128,3), activation='relu'),
tf.keras.layers.MaxPool2D(pool_size=(2,2)),
tf.keras.layers.Conv2D(filters=20, kernel_size=5, activation='relu'),
tf.keras.layers.Dropout(rate=0.3),
tf.keras.layers.MaxPool2D(pool_size=(2,2)),

tf.keras.layers.Flatten(),
tf.keras.layers.Dense(320, activation='relu'),
tf.keras.layers.Dense(50, activation='relu'),
tf.keras.layers.Dropout(0.2),
tf.keras.layers.Dense(n_classes, activation='softmax')
]
)
    return model

model = create_tf_model()
model.build()
```

[5]

```
2022-07-02 19:04:26.155914: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcuda.so.1
```

Figure 38: Building the Model



Compute instance: ChestXRay-Compute - Running

Python 3.8 - AzureML

ChestXRay-Compute · Kernel idle CPU 0% RAM 3% Last saved a few seconds ago Python 3.8 - AzureML

```
model.summary()
```

[6]

```
Model: "sequential"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
conv2d (Conv2D)             (None, 124, 124, 10)      760
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 62, 62, 10)       0
_________________________________________________________________
conv2d_1 (Conv2D)           (None, 58, 58, 20)        5020
_________________________________________________________________
dropout (Dropout)           (None, 58, 58, 20)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 29, 29, 20)       0
_________________________________________________________________
flatten (Flatten)           (None, 16820)             0
```

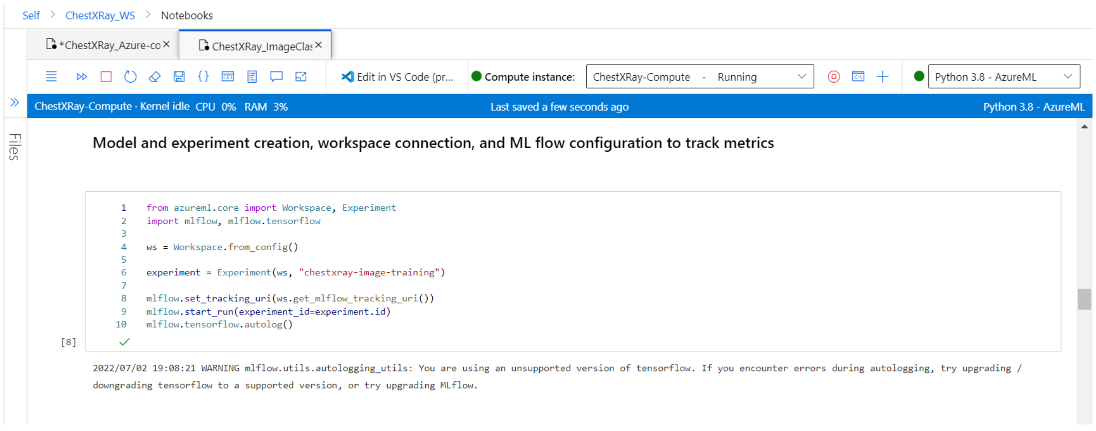
```
## Taking Adam as our model optimizer since it implements gradient descent algorithm
model.compile(optimizer="adam", loss='categorical_crossentropy', metrics=["accuracy"])
```

[7]

Figure 39: Build the Model

6.3.7 Workspace Connection and ML Flow Configuration

To measure metrics and log model artifacts, we utilized MLflow autologging.



Compute instance: ChestXRay-Compute - Running

Python 3.8 - AzureML

ChestXRay-Compute · Kernel idle CPU 0% RAM 3% Last saved a few seconds ago Python 3.8 - AzureML

Model and experiment creation, workspace connection, and ML flow configuration to track metrics

```
from azureml.core import Workspace, Experiment
import mlflow, mlflow.tensorflow

ws = Workspace.from_config()

experiment = Experiment(ws, "chestxray-image-training")

mlflow.set_tracking_uri(ws.get_mlflow_tracking_uri())
mlflow.start_run(experiment_id=experiment.id)
mlflow.tensorflow.autolog()
```

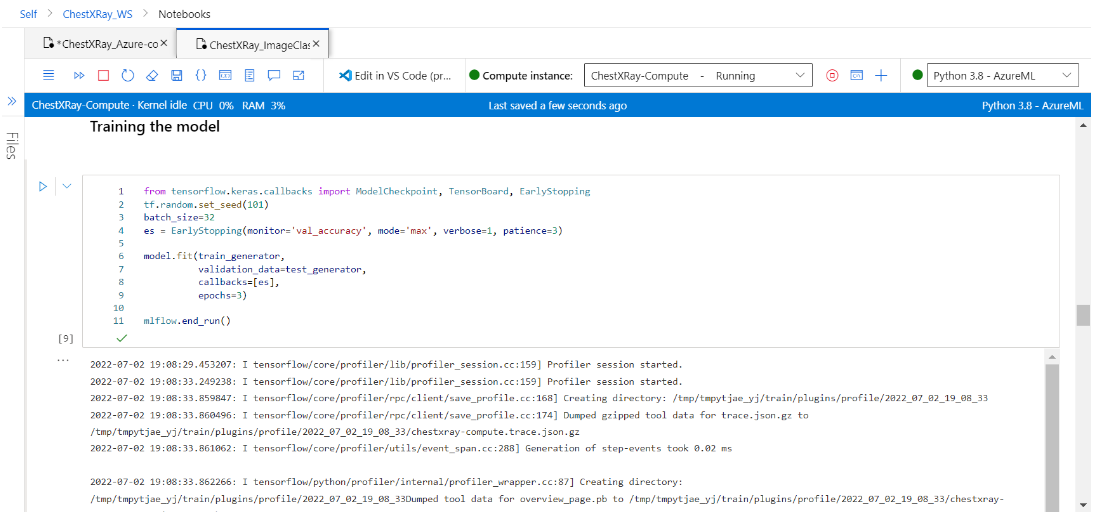
[8]

```
2022/07/02 19:08:21 WARNING mlflow.utils.autologging_utils: You are using an unsupported version of tensorflow. If you encounter errors during autologging, try upgrading /
downgrading tensorflow to a supported version, or try upgrading MLflow.
```

Figure 40: Workspace Connection and ML Flow Configuration

6.3.8 Train Model

We must develop a deep learning model and a training script that can read images in order to train the model.


Self > ChestXRay_WS > Notebooks
*ChestXRay_Azure-co
ChestXRay_ImageClas
Edit in VS Code (pr...
Compute instance:
ChestXRay-Compute - Running
Python 3.8 - AzureML
ChestXRay-Compute · Kernel idle CPU 0% RAM 3%
Last saved a few seconds ago
Files

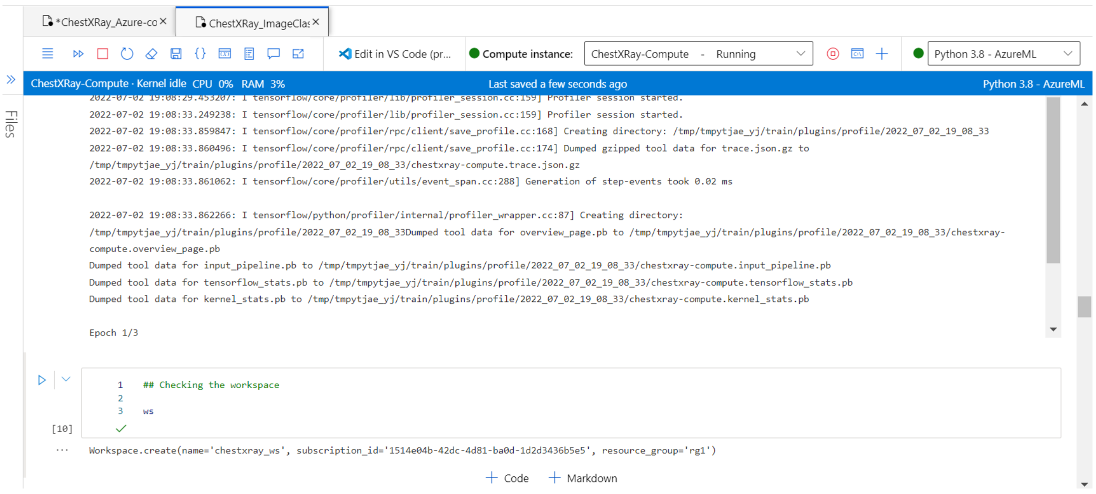
Training the model

```
from tensorflow.keras.callbacks import ModelCheckpoint, TensorBoard, EarlyStopping
tf.random.set_seed(101)
batch_size=32
es = EarlyStopping(monitor='val_accuracy', mode='max', verbose=1, patience=3)

model.fit(train_generator,
          validation_data=test_generator,
          callbacks=[es],
          epochs=3)

mlflow.end_run()
```

[9]

```
2022-07-02 19:08:29.453207: I tensorflow/core/profiler/lib/profiler_session.cc:159] Profiler session started.
2022-07-02 19:08:33.249238: I tensorflow/core/profiler/lib/profiler_session.cc:159] Profiler session started.
2022-07-02 19:08:33.859847: I tensorflow/core/profiler/rpc/client/save_profile.cc:168] Creating directory: /tmp/tmpytjae_yj/train/plugins/profile/2022_07_02_19_08_33
2022-07-02 19:08:33.860496: I tensorflow/core/profiler/rpc/client/save_profile.cc:174] Dumped gzipped tool data for trace.json.gz to
/tmp/tmpytjae_yj/train/plugins/profile/2022_07_02_19_08_33/chestxray-compute.trace.json.gz
2022-07-02 19:08:33.861062: I tensorflow/core/profiler/utils/event_span.cc:288] Generation of step-events took 0.02 ms

2022-07-02 19:08:33.862266: I tensorflow/python/profiler/internal/profiler_wrapper.cc:87] Creating directory:
/tmp/tmpytjae_yj/train/plugins/profile/2022_07_02_19_08_33Dumped tool data for overview_page.pb to /tmp/tmpytjae_yj/train/plugins/profile/2022_07_02_19_08_33/chestxray-
```

Figure 41: Train the Model



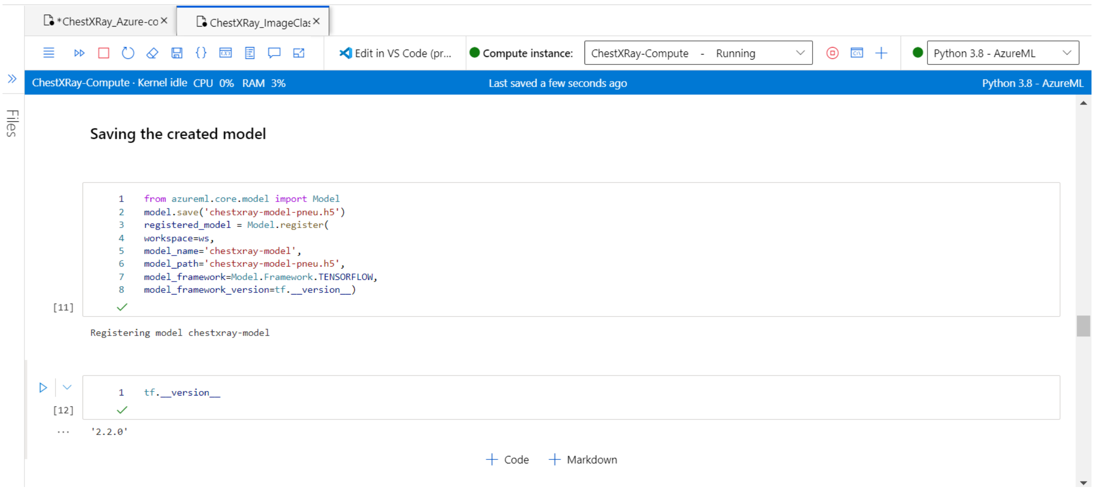
```
2022-07-02 19:08:33.249238: I tensorflow/core/profiler/lib/profiler_session.cc:159] Profiler session started.
2022-07-02 19:08:33.859847: I tensorflow/core/profiler/rpc/client/save_profile.cc:168] Creating directory: /tmp/tmpytjae_yj/train/plugins/profile/2022_07_02_19_08_33
2022-07-02 19:08:33.860496: I tensorflow/core/profiler/rpc/client/save_profile.cc:174] Dumped gzipped tool data for trace.json.gz to
/tmp/tmpytjae_yj/train/plugins/profile/2022_07_02_19_08_33/chestxray-compute.trace.json.gz
2022-07-02 19:08:33.861062: I tensorflow/core/profiler/utils/event_span.cc:288] Generation of step-events took 0.02 ms

2022-07-02 19:08:33.862266: I tensorflow/python/profiler/internal/profiler_wrapper.cc:87] Creating directory:
/tmp/tmpytjae_yj/train/plugins/profile/2022_07_02_19_08_33Dumped tool data for overview_page.pb to /tmp/tmpytjae_yj/train/plugins/profile/2022_07_02_19_08_33/chestxray-
compute.overview_page.pb
Dumped tool data for input_pipeline.pb to /tmp/tmpytjae_yj/train/plugins/profile/2022_07_02_19_08_33/chestxray-compute.input_pipeline.pb
Dumped tool data for tensorflow_stats.pb to /tmp/tmpytjae_yj/train/plugins/profile/2022_07_02_19_08_33/chestxray-compute.tensorflow_stats.pb
Dumped tool data for kernel_stats.pb to /tmp/tmpytjae_yj/train/plugins/profile/2022_07_02_19_08_33/chestxray-compute.kernel_stats.pb

Epoch 1/3
```

```
## Checking the workspace

ws
```

[10]

```
Workspace.create(name='chestxray_ws', subscription_id='1514e04b-42dc-4d81-ba0d-1d2d3436b5e5', resource_group='rg1')
```

Figure 42: Train the Model



Saving the created model

```
from azureml.core.model import Model
model.save('chestxray-model-pneu.h5')
registered_model = Model.register(
workspace=ws,
model_name='chestxray-model',
model_path='chestxray-model-pneu.h5',
model_framework=Model.Framework.TENSORFLOW,
model_framework_version=tf.__version__)
```

[11]

```
Registering model chestxray-model
```

```
tf.__version__
```

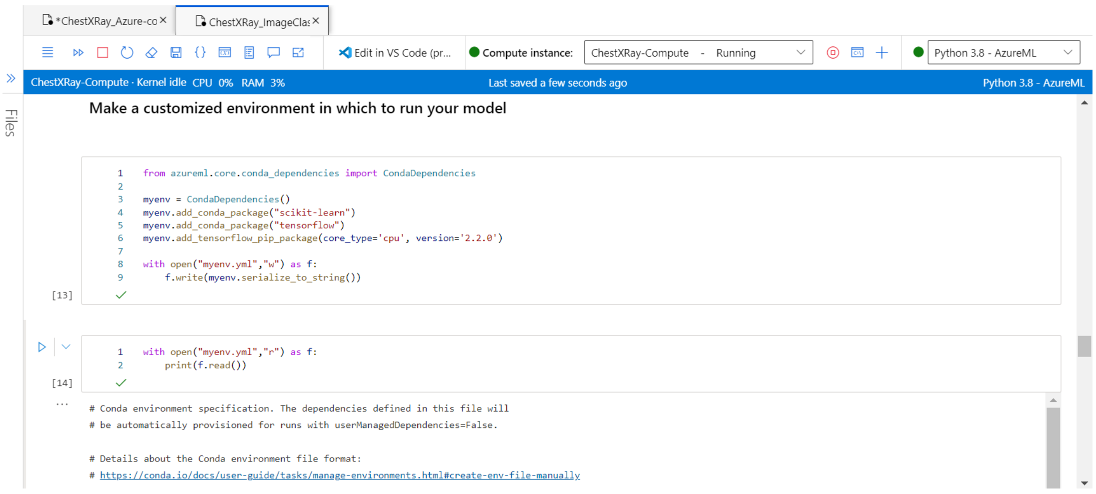
[12]

```
'2.2.0'
```

Figure 43: Save the Created Model

6.3.9 Create Environment and Container Service

Create Environment to Run the Model In and a container service to contain the model. An experiment is a collection of multiple runs from a single script or piece of code. Under such experiment, information for the run is saved. If the name does not exist when we submit an experiment, we will see numerous tabs including metrics, logs, explanations, and so on.


*ChestXRay_Azure-co
ChestXRay_ImageClas
Edit in VS Code (pr...
Compute instance: ChestXRay-Compute - Running
Python 3.8 - AzureML
ChestXRay-Compute · Kernel idle CPU 0% RAM 3%
Last saved a few seconds ago
Python 3.8 - AzureML
Files

Make a customized environment in which to run your model

```
from azureml.core.conda_dependencies import CondaDependencies

myenv = CondaDependencies()
myenv.add_conda_package("scikit-learn")
myenv.add_conda_package("tensorflow")
myenv.add_tensorflow_pip_package(core_type='cpu', version='2.2.0')

with open("myenv.yml","w") as f:
    f.write(myenv.serialize_to_string())
```

[13]

```
with open("myenv.yml","r") as f:
    print(f.read())
```

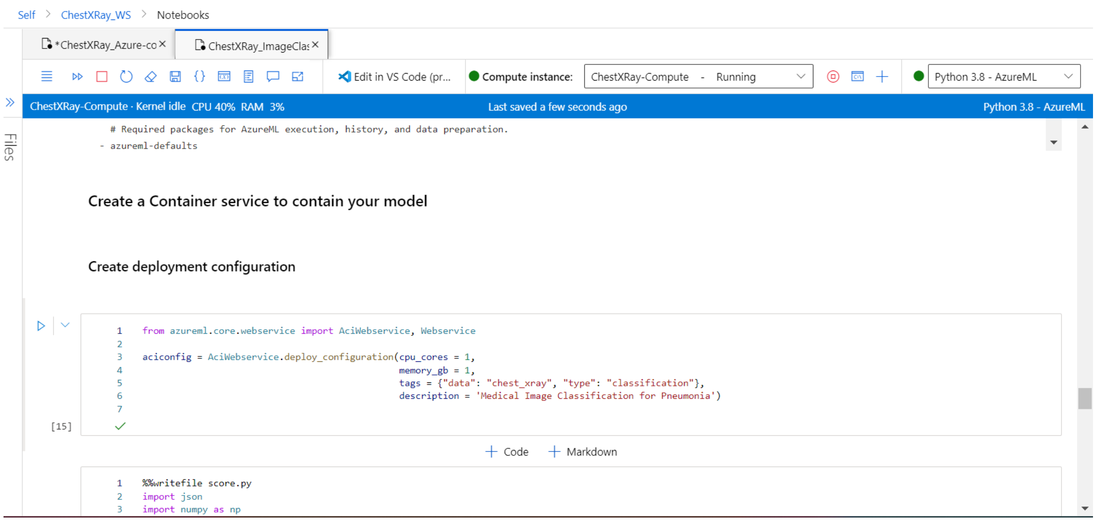
[14]

```
# Conda environment specification. The dependencies defined in this file will
# be automatically provisioned for runs with userManagedDependencies=False.

# Details about the Conda environment file format:
# https://conda.io/docs/user-guide/tasks/manage-environments.html#create-env-file-manually
```

Figure 44: Environment Creation

6.3.10 Deploy Model

Here, we deployed our model so that an application may consume (infer) the model over REST. The code below creates a curated environment that includes all of the requirements needed to host the model (for example, the packages like scikit-learn). We also generated a deployment configuration that defines how much computation is necessary to host the model. In this scenario, the compute will have one CPU and one gigabyte of RAM.


Self > ChestXRay_WS > Notebooks
*ChestXRay_Azure-co
ChestXRay_ImageClas
Edit in VS Code (pr...
Compute instance: ChestXRay-Compute - Running
Python 3.8 - AzureML
ChestXRay-Compute · Kernel idle CPU 40% RAM 3%
Last saved a few seconds ago
Python 3.8 - AzureML
Files
+ Code
+ Markdown

```
    # Required packages for AzureML execution, history, and data preparation.
  - azureml-defaults
```

Create a Container service to contain your model

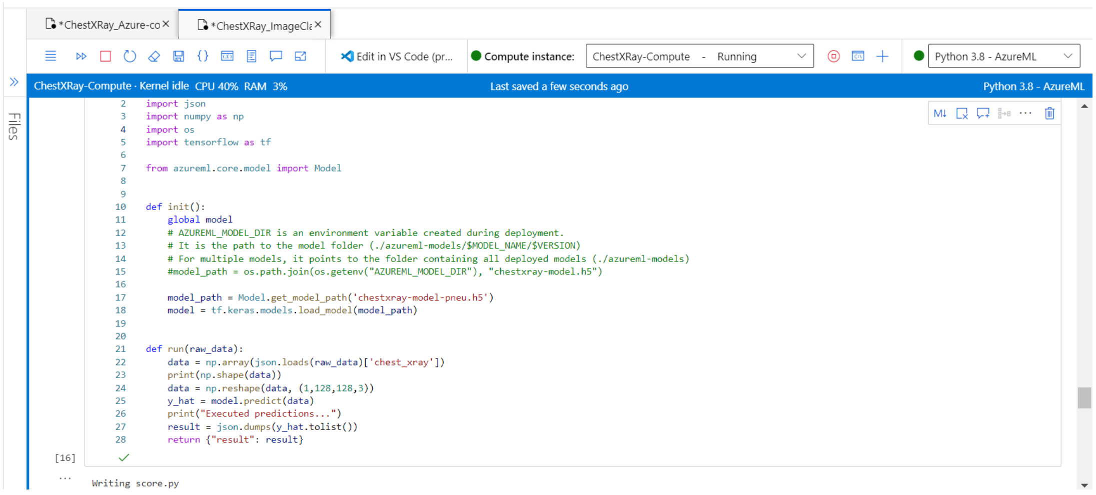
Create deployment configuration

```
from azureml.core.webservice import AciWebservice, Webservice

aciconfig = AciWebservice.deploy_configuration(cpu_cores = 1,
                                               memory_gb = 1,
                                               tags = {"data": "chest_xray", "type": "classification"},
                                               description = 'Medical Image Classification for Pneumonia')

```

[15]

```
%%writefile score.py
import json
import numpy as np
```

Figure 45: Deployment Configuration Creation

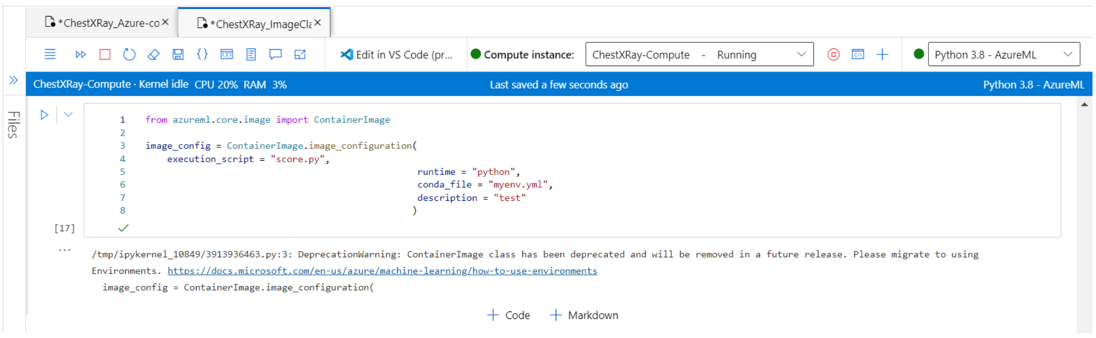


```
import json
import numpy as np
import os
import tensorflow as tf

from azureml.core.model import Model


def init():
    global model
    # AZUREML_MODEL_DIR is an environment variable created during deployment.
    # It is the path to the model folder (./azureml-models/$MODEL_NAME/$VERSION)
    # For multiple models, it points to the folder containing all deployed models (./azureml-models)
    #model_path = os.path.join(os.getenv("AZUREML_MODEL_DIR"), "chestxray-model.h5")

    model_path = Model.get_model_path('chestxray-model-pneu.h5')
    model = tf.keras.models.load_model(model_path)


def run(raw_data):
    data = np.array(json.loads(raw_data)['chest_xray'])
    print(np.shape(data))
    data = np.reshape(data, (1,128,128,3))
    y_hat = model.predict(data)
    print("Executed predictions...")
    result = json.dumps(y_hat.tolist())
    return {"result": result}
```

[16]

```
Writing score.py
```

Figure 46: Deployment Configuration Creation



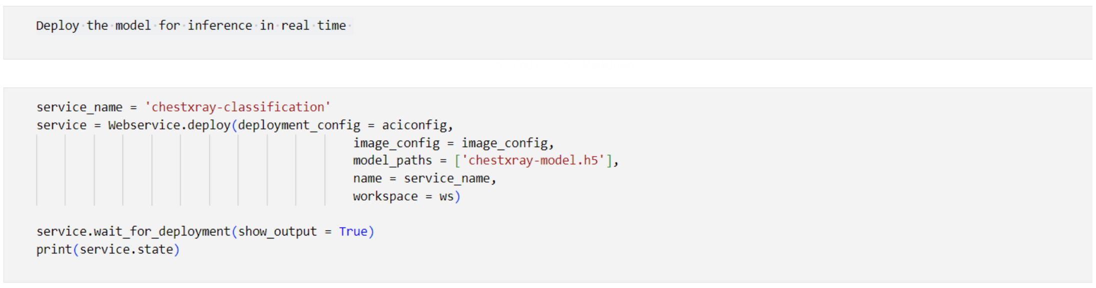
```
from azureml.core.image import ContainerImage

image_config = ContainerImage.image_configuration(
    execution_script = "score.py",
                                                 runtime = "python",
                                                 conda_file = "myenv.yml",
                                                 description = "test"
                                                )
```

[17]

```
/tmp/ipykernel_10849/3913936463.py:3: DeprecationWarning: ContainerImage class has been deprecated and will be removed in a future release. Please migrate to using
Environments. https://docs.microsoft.com/en-us/azure/machine-learning/how-to-use-environments
  image_config = ContainerImage.image_configuration(
```

Figure 47: Container Image Creation

The code below deploys the model to Azure Container Instance.



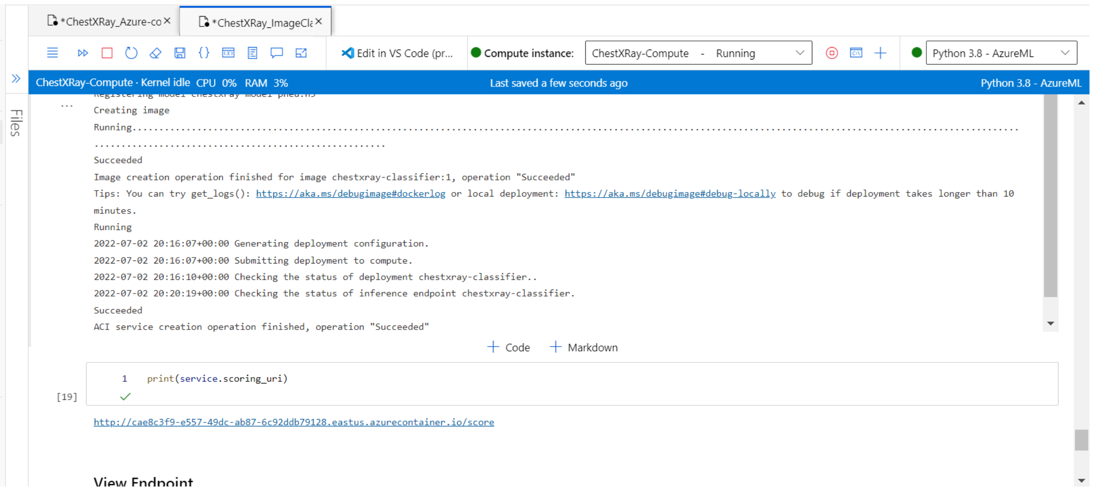
Deploy the model for inference in real time

```
service_name = 'chestxray-classification'
service = Webservice.deploy(deployment_config = aciconfig,
                                             image_config = image_config,
                                             model_paths = ['chestxray-model.h5'],
                                             name = service_name,
                                             workspace = ws)

service.wait_for_deployment(show_output = True)
print(service.state)
```

Figure 48: Model Deployment

6.3.11 Endpoint

When the model has been successfully deployed, go to Endpoints in the left-hand menu of Azure Machine Learning Studio to view the endpoint. You may view the endpoint's status (healthy or unhealthy), logs, and consumption (how applications can consume the model).



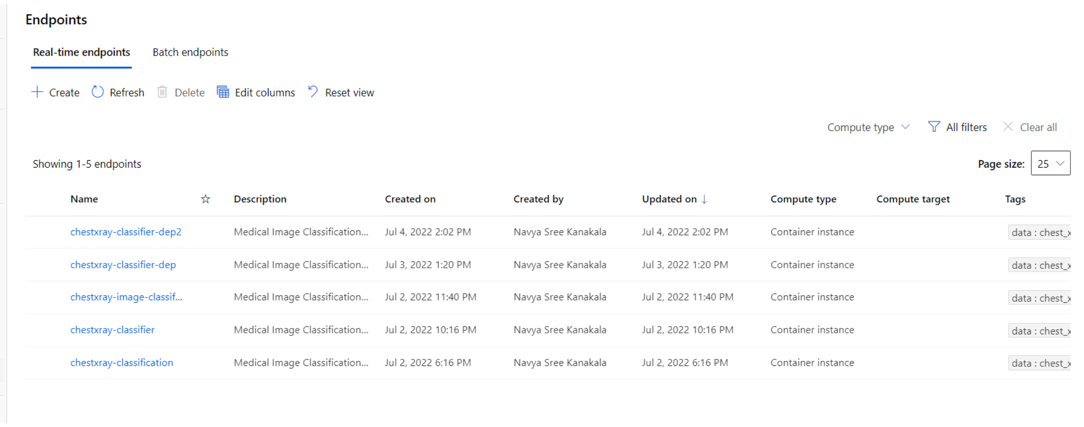
```
Creating image
Running.................................................................................................................................
..................................................
Succeeded
Image creation operation finished for image chestxray-classifier:1, operation "Succeeded"
Tips: You can try get_logs(): https://aka.ms/debugimage#dockerlog or local deployment: https://aka.ms/debugimage#debug-locally to debug if deployment takes longer than 10
minutes.
Running
2022-07-02 20:16:07+00:00 Generating deployment configuration.
2022-07-02 20:16:07+00:00 Submitting deployment to compute.
2022-07-02 20:16:10+00:00 Checking the status of deployment chestxray-classifier..
2022-07-02 20:20:19+00:00 Checking the status of inference endpoint chestxray-classifier.
Succeeded
ACI service creation operation finished, operation "Succeeded"
```

```
[19]  print(service.scoring_uri)
```

```
http://cae8c3f9-e557-49dc-ab87-6c92ddb79128.eastus.azurecontainer.io/score
```

View Endpoint

Figure 49: Endpoint Created

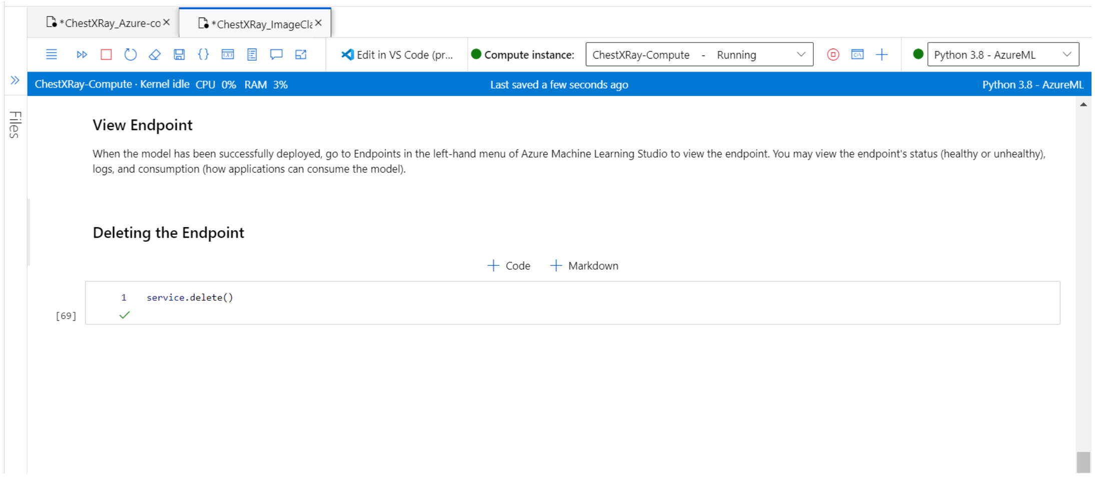


Showing 1-5 endpoints

Name	Description	Created on	Created by	Updated on ↓	Compute type	Compute target	Tags
chestxray-classifier-dep2	Medical Image Classification...	Jul 4, 2022 2:02 PM	Navya Sree Kanakala	Jul 4, 2022 2:02 PM	Container instance		data : chest_x
chestxray-classifier-dep	Medical Image Classification...	Jul 3, 2022 1:20 PM	Navya Sree Kanakala	Jul 3, 2022 1:20 PM	Container instance		data : chest_x
chestxray-image-classif...	Medical Image Classification...	Jul 2, 2022 11:40 PM	Navya Sree Kanakala	Jul 2, 2022 11:40 PM	Container instance		data : chest_x
chestxray-classifier	Medical Image Classification...	Jul 2, 2022 10:16 PM	Navya Sree Kanakala	Jul 2, 2022 10:16 PM	Container instance		data : chest_x
chestxray-classification	Medical Image Classification...	Jul 2, 2022 6:16 PM	Navya Sree Kanakala	Jul 2, 2022 6:16 PM	Container instance		data : chest_x

Figure 50: Endpoints

If we won't be utilizing this model any longer, remove the Model service by using code below:



View Endpoint

When the model has been successfully deployed, go to Endpoints in the left-hand menu of Azure Machine Learning Studio to view the endpoint. You may view the endpoint's status (healthy or unhealthy), logs, and consumption (how applications can consume the model).

Deleting the Endpoint

```
[69]  service.delete()
```

Figure 51: Deleting Endpoint

Stop the compute instance by choosing the "Stop compute" option next to the Compute dropdown if we wish to further manage costs. Then, the next time you need it, restart the compute instance. Remove all computational resources and your Azure Machine Learning workspace.

7 Conclusion

The resulting highly accurate model indicates that this AI system has the capability to efficiently learn from increasingly complex images with a high degree of generalization using a relatively small repository of data. This transfer learning framework offers a convincing system for further investigation and analysis in biomedical imaging as well as more broadly applied use to an automated community-based AI system for the diagnosis and triage of common human diseases by demonstrating effectiveness with multiple imaging modalities and with a wide range of pathology.

This project clearly demonstrates how users can use Amazon SageMaker and AzureML to easily build, train, and deploy machine learning models. This could make screening programs and referral systems in all of medicine more effective, especially in remote or underdeveloped areas, with a consequent broad clinical and public health impact.

To utilize AWS AI products, you must be proficient in coding and data science. When constructing ML models, SageMaker gives you maximum control and freedom. Any concept may be realized, but to properly exploit AWS features, you must be familiar with Jupyter Notebook and Python. Azure ML Studio, on the other hand, is built on a codeless experience. You do not need to be an expert in complex data science techniques or know how to write in Python. The service is intended for data analysts that prefer graphical element display and a straightforward interface.

This document discussed two popular platforms for developing machine learning and artificial intelligence. We examined their features, similarities, and differences. Both are excellent choices for developing and deploying machine learning models, but each has advantages and disadvantages. AWS Sagemaker is an excellent platform for developing simple models and deploying them in the cloud with minimal configuration. However, for predictive analytics, Azure ML may be a more versatile option.

To conclude, the approach we developed on AWS is independent of any cloud or local platform. It does not utilize any pre-built sagemaker libraries or frameworks, but the code on Azure includes Azure ML libraries, making it impossible to use the same code on other platforms. Our model detects Pnuemonia quite effectively, but standard detection is not as successful. Because our purpose is to forecast pneumonia, which is being accomplished over here. We can enhance the model by leveraging current models or by developing new ones. We can also use the fast or flask APIs to expand our AWS model. With all of the hype surrounding machine learning, it is clear that the future will be data-driven. Machine learning capabilities can be used by organizations to solve current business challenges and prepare for future business opportunities.

References

[1] T.M. Mitchell, J.G. Carbonell, and R.S. Michalski. *Machine Learning: A Guide to Current Research.* The Springer International Series in Engineering and Computer Science. Springer US, 2012. Accessed on 09.07.2022.

[2] K. Hwang. *Cloud Computing for Machine Learning and Cognitive Applications.* The MIT Press. MIT Press, 2017. Accessed on 09.07.2022.

[3] Amazon SageMaker Developer Guide. Accessed on 09.07.2022.

[4] Amazon Simple Storage Service User Guide. Accessed on 09.07.2022.

[5] AWS Identity and Access Management User Guide. Accessed on 09.07.2022.

[6] The Jupyter Notebook Documentation. Accessed on 09.07.2022.

[7] Azure Machine Learning.

[8] Jafar Alzubi, Anand Nayyar, and Akshi Kumar. Machine Learning from Theory to Algorithms: An Overview. 2018. Accessed on 09.07.2022.

[9] Image classification on Azure. Accessed on 09.07.2022.