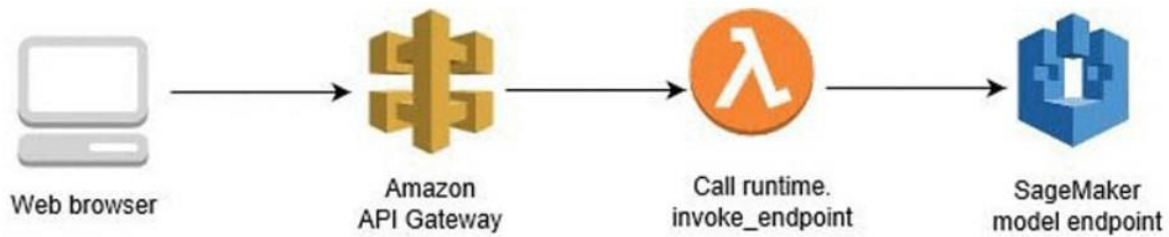


Assignment 4 Report

Architecture diagram:



Steps:

1. First, I downloaded the mnist dataset from the following link:
<https://www.kaggle.com/datasets/oddrational/mnist-in-csv>
2. This dataset consists of 785 columns where the first column represents the label of the image and the next 784 columns represents the pixels values.
3. Then I created a notebook instance in amazon sagemaker and created a s3 bucket to store the training data.

```
In [2]: bucket_name = 'hk264208' # <--- CHANGE THIS VARIABLE TO A UNIQUE NAME FOR YOUR BUCKET
s3 = boto3.resource('s3')
try:
    if my_region == 'us-east-1':
        s3.create_bucket(Bucket=bucket_name)
    else:
        s3.create_bucket(Bucket=bucket_name, CreateBucketConfiguration={ 'LocationConstraint': my_region })
    print('S3 bucket created successfully')
except Exception as e:
    print('S3 error: ',e)
```

4. Then I read the mnist training data.

```
In [3]: model_data = pd.read_csv('mnist_train.csv')
```

5. Next, I wrote a function to upload the training data to s3 bucket.

```
In [13]: import io

s3_resource = boto3.Session().resource('s3')
prefix='mnistdata'
train_file = 'mnist_train.csv'
def upload_s3_csv(filename, folder, dataframe):
    csv_buffer = io.StringIO()
    dataframe.to_csv(csv_buffer, header=False, index=False)
    s3_resource.Bucket(bucket_name).Object(os.path.join(prefix, folder, filename)).put(Body=csv_buffer.getvalue())
```

```
In [14]: upload_s3_csv(train_file, 'train', train)
```

6. Then I got the KNN model with help of “sagemaker.estimator.Estimator” module.

```
In [103]: knn = sagemaker.estimator.Estimator(
    get_image_uri(boto3.Session().region_name, "knn"),
    get_execution_role(),
    instance_count=1,
    instance_type="ml.m4.xlarge",
    output_path="s3://{}/{}/output/".format(bucket_name,prefix),
    sagemaker_session=sagemaker.Session()
)
```

The method `get_image_uri` has been renamed in `sagemaker>=2`.
See: <https://sagemaker.readthedocs.io/en/stable/v2.html> for details.

7. Next, I set the hyperparameters for the knn model.

```
In [131]: knn.set_hyperparameters(feature_dim= 784, k= 10, sample_size= 200000, predictor_type= "classifier")
```

8. To train the knn model we need to provide it with the training data which I uploaded it in s3 bucket in step 5.

```
In [102]: s3_input_train = sagemaker.inputs.TrainingInput(s3_data="s3://{}/{}/train/".format(bucket_name, prefix, train_file),
    content_type='text/csv')
```

9. Next, I trained the knn model.

```
In [132]: knn.fit({'train': s3_input_train})

[11/19/2022 22:34:22 INFO 139728985028416] ...Finished training index in 0 second(s)
[11/19/2022 22:34:22 INFO 139728985028416] Adding data to index...
[11/19/2022 22:34:22 INFO 139728985028416] ...Finished adding data to index in 0 second(s)
#metrics {"StartTime": 1668897256.447919, "EndTime": 1668897262.3561966, "Dimensions": {"Algorithm": "AWS/KNN", "Host": "algo
-1", "Operation": "training"}, "Metrics": {"initialize.time": {"sum": 81.90488815307617, "count": 1, "min": 81.9048881530761
7, "max": 81.90488815307617}, "epochs": {"sum": 1.0, "count": 1, "min": 1, "max": 1}, "update.time": {"sum": 5153.24544906616
2, "count": 1, "min": 5153.245449066162, "max": 5153.245449066162}, "finalize.time": {"sum": 423.5258102416992, "count": 1,
"min": 423.5258102416992, "max": 423.5258102416992}, "model.serialize.time": {"sum": 144.70338821411133, "count": 1, "min": 1
44.70338821411133, "max": 144.70338821411133}}}
[11/19/2022 22:34:22 INFO 139728985028416] Test data is not provided.
#metrics {"StartTime": 1668897262.3563538, "EndTime": 1668897262.3652694, "Dimensions": {"Algorithm": "AWS/KNN", "Host": "alg
o-1", "Operation": "training"}, "Metrics": {"setuptime": {"sum": 23.123502731323242, "count": 1, "min": 23.123502731323242,
"max": 23.123502731323242}, "totaltime": {"sum": 6892.109155654907, "count": 1, "min": 6892.109155654907, "max": 6892.1091556
54907}}}

2022-11-19 22:34:52 Uploading - Uploading generated training model
2022-11-19 22:34:52 Completed - Training job completed
Training seconds: 283
Billable seconds: 283
```

10. After training the model I deployed the model with an endpoint name.

```
In [106]: EndpointName = 'hk45FirstEndpoint'

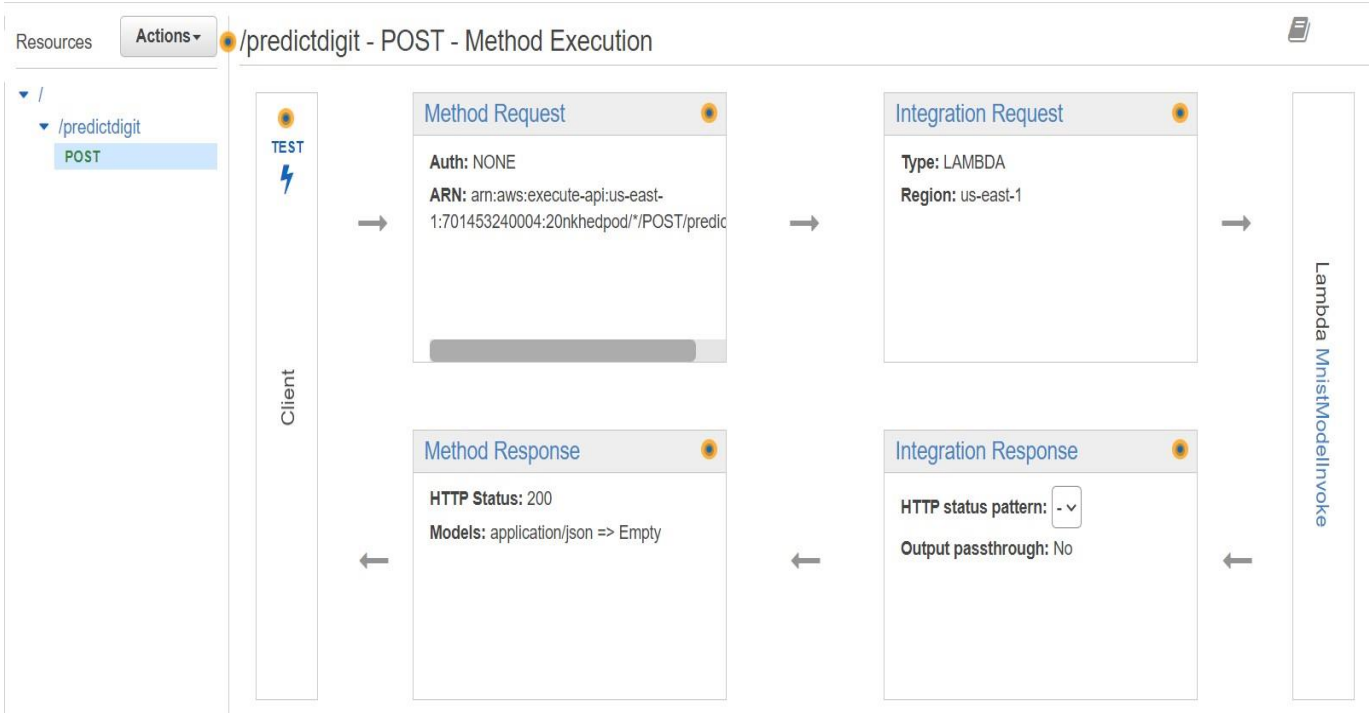
In [133]: knn_predictor = knn.deploy(initial_instance_count=1, serializer = sagemaker.serializers.CSVSerializer(),
                                     instance_type='ml.m4.xlarge', endpoint_name=EndpointName )

-----!
```

11. Next, I created a lambda function with an IAM Role that allows the lambda function to access the model endpoint.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": "sagemaker:InvokeEndpoint",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "logs:CreateLogGroup",
      "Resource": "arn:aws:logs:us-east-1:701453240004:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:701453240004:log-group:/aws/lambda/MnistModelInvoke:*"
      ]
    }
  ]
}
```

12.Next, I created a REST API Post method that Integrated with the lambda function I created.



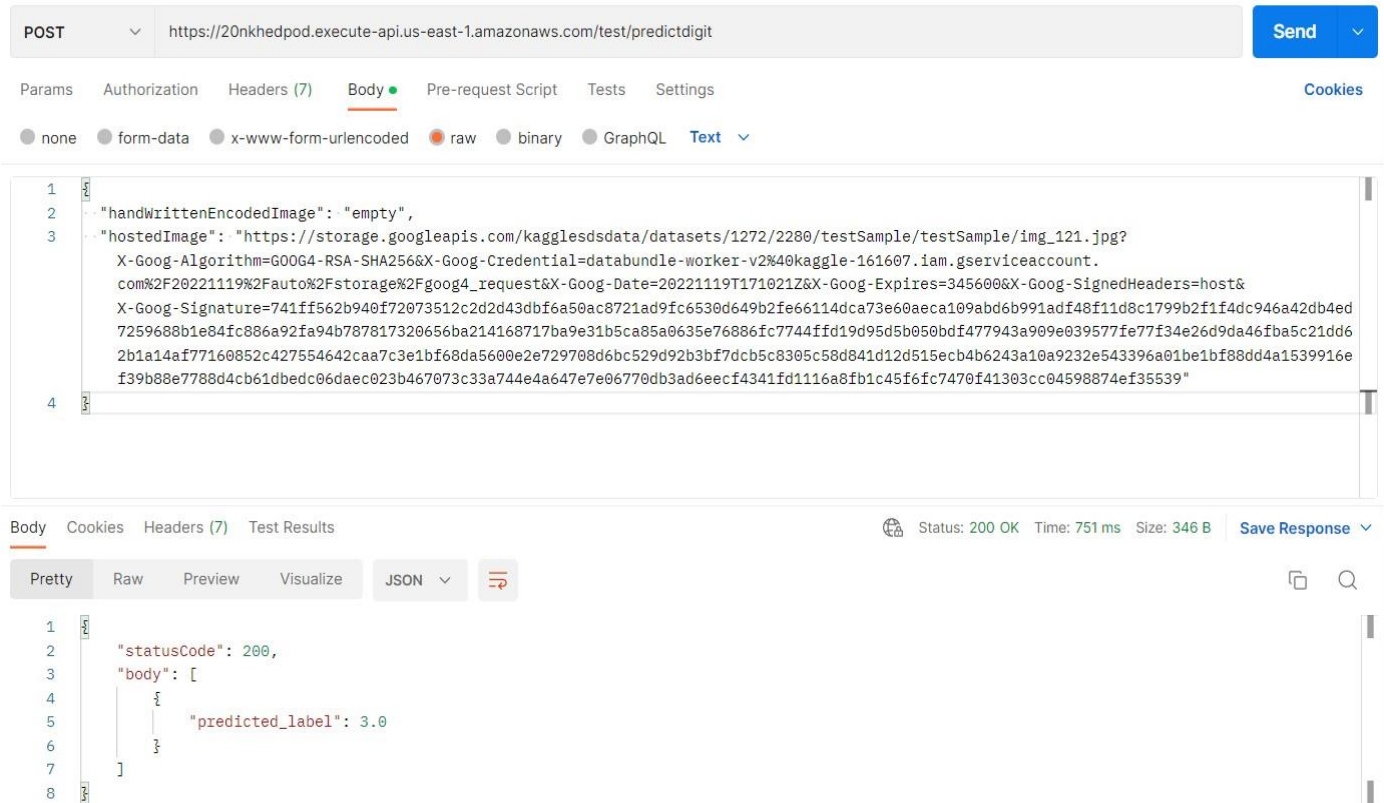
13. Since the lambda function doesn't contain libraries like numpy, PIL, requests I added these libraries into lambda function by creating layers.

Lambda > Layers

Layers (3) Last fetched now Create layer

Name	Version	Compatible runtimes	Compatible architectures
Pillow	1	python3.7	-
pandas	2	python3.7	-
requests	1	python3.7	-

14. Next, I sent a POST request with API Link I created that contains a body which has a JSON object with two properties “handWrittenEncodedImage” and “hostedImage”.



The screenshot shows a REST client interface. The top bar indicates a POST request to the URL `https://20nkhedpod.execute-api.us-east-1.amazonaws.com/test/predictdigit`. The 'Body' tab is selected, showing a JSON object with two properties: `"handWrittenEncodedImage": "empty"` and `"hostedImage": "https://storage.googleapis.com/kagglestdatasets/1272/2280/testSample/testSample/img_121.jpg?X-Goog-Algorithm=G00G4-RSA-SHA256&X-Goog-Credential=databundle-worker-v2%40kaggle-161607.iam.gserviceaccount.com%2F20221119%2Fauto%2Fstorage%2Fgoog4_request&X-Goog-Date=20221119T171021Z&X-Goog-Expires=345600&X-Goog-SignedHeaders=host&X-Goog-Signature=741ff562b940f72073512c2d2d43dbf6a50ac8721ad9fc6530d649b2fe66114dca73e60aeca109abd6b991adf48f11d8c1799b2f1f4dc946a42db4ed7259688b1e84fc886a92fa94b787817320656ba214168717ba9e31b5ca85a0635e76886fc7744ffd19d95d5b050bdf477943a909e039577fe77f34e26d9da46fba5c21dd62b1a14af77160852c427554642caa7c3e1bf68da5600e2e729708d6bc529d92b3bf7dcb5c8305c58d841d12d515ecb4b6243a10a9232e543396a01be1bf88dd4a1539916ef39b88e7788d4cb61dbedc06daec023b467073c33a744e4a647e7e06770db3ad6e6c4341fd1116a8fb1c45f6fc7470f41303cc04598874ef35539"`. The bottom section shows the response in JSON format: `{ "statusCode": 200, "body": [{ "predicted_label": 3.0 }] }`. The status is 200 OK, time is 751 ms, and size is 346 B.

15. When sending the POST request only one property has a value and the other property has a value “empty” because the model predicts one image at a time.

16. “handWrittenEncodedImage” property will take a base64 encoded string value and “hostedImage” property will take a hyperlink that contains an image hosted by kaggle website.

17. “handWrittenEncodedImage” property value will be decoded in lambda function and some image processing (resizing to 28X28 size and converting it into greyscale) will be done and then it will be passed to model to predict the label.

```
dec_img_str = base64.b64decode(str(handWriEncImg))
img = Image.open(BytesIO(dec_img_str))
img = img.resize((28, 28), Image.ANTIALIAS)
img = ImageOps.invert(ImageOps.grayscale(img))
img = np.array(img.getdata(), dtype = np.uint8).reshape(28, 28)
```

18. For “hostedImage” property, in lambda function its value will be passed to requests.get() function to get the image and then it will be passed to the model to get the predicted label.

```
response = requests.get(hostImg)
img = Image.open(BytesIO(response.content))
img = np.array(img.getdata(), dtype = np.uint8).reshape(img.size[0], img.size[1])
```

19. I attached the all the lambda function code in github, please do check it.

Hari Krishna Vinjam

People soft id: 2199727

Email id: hvinjam@cougarnet.uh.edu