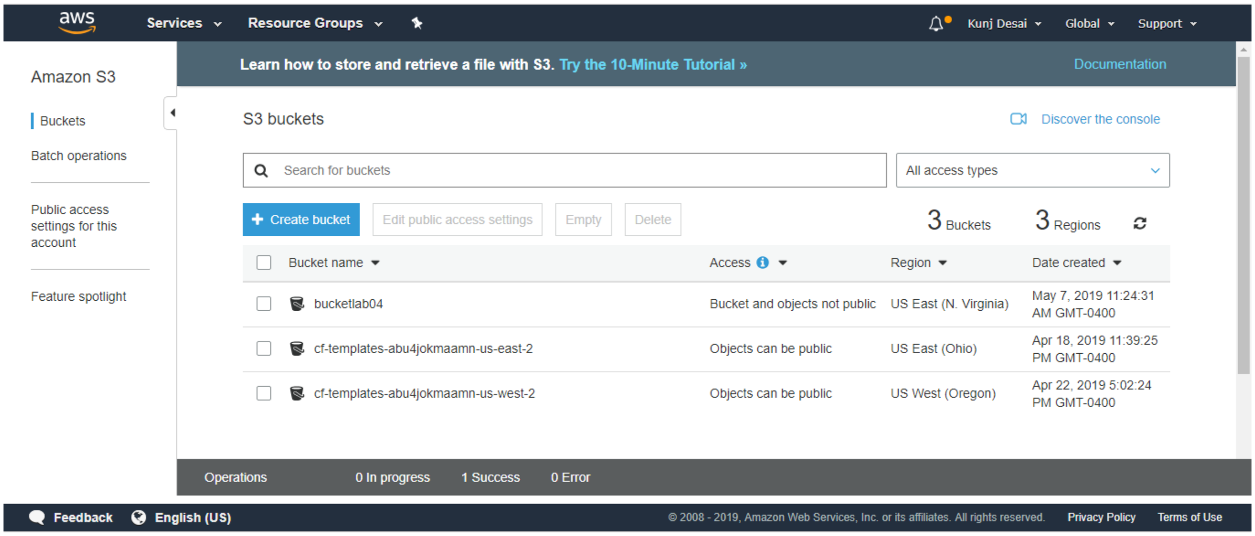
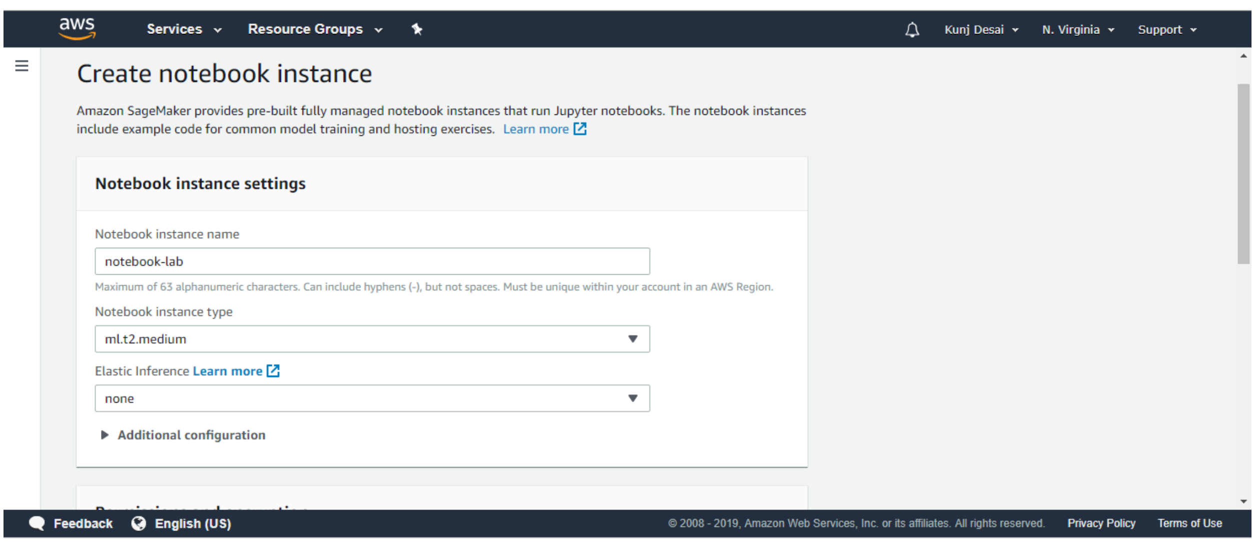
Lab Assignment 5

➢ **These are the following twenty steps we need to follow for achieving the goal of the assignment:**

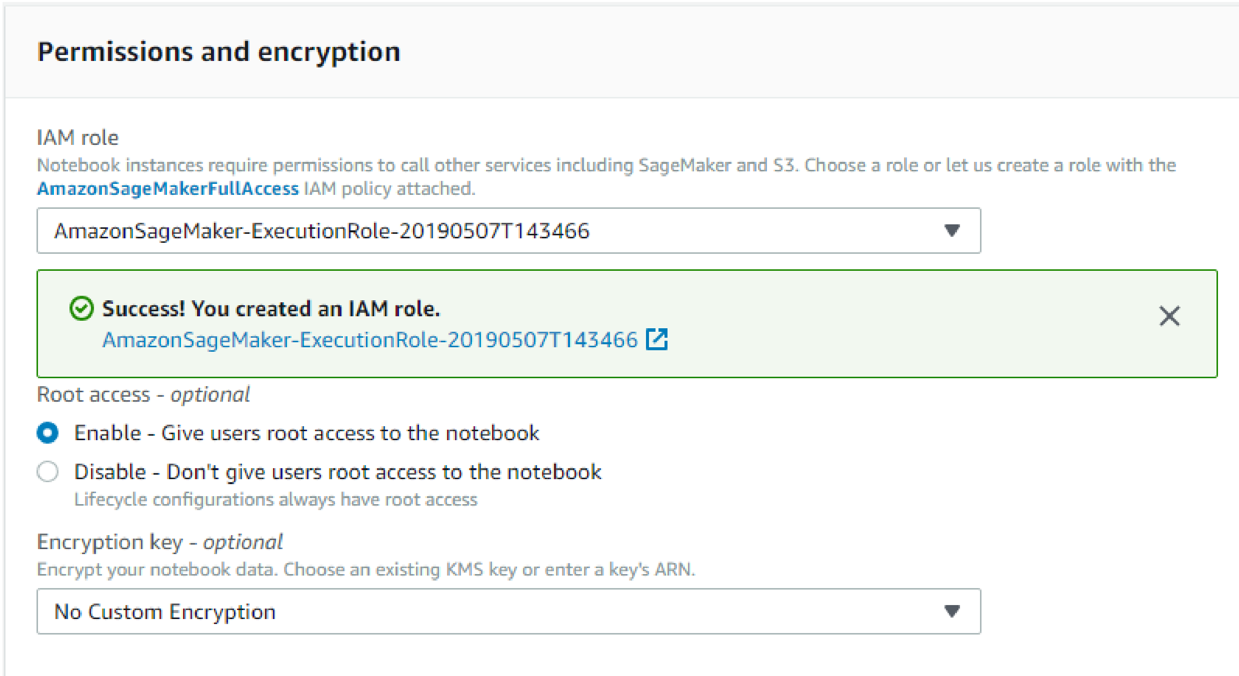
1. After logging in with the AWS console, I created S3 bucket with the name "**bucketlab04**".



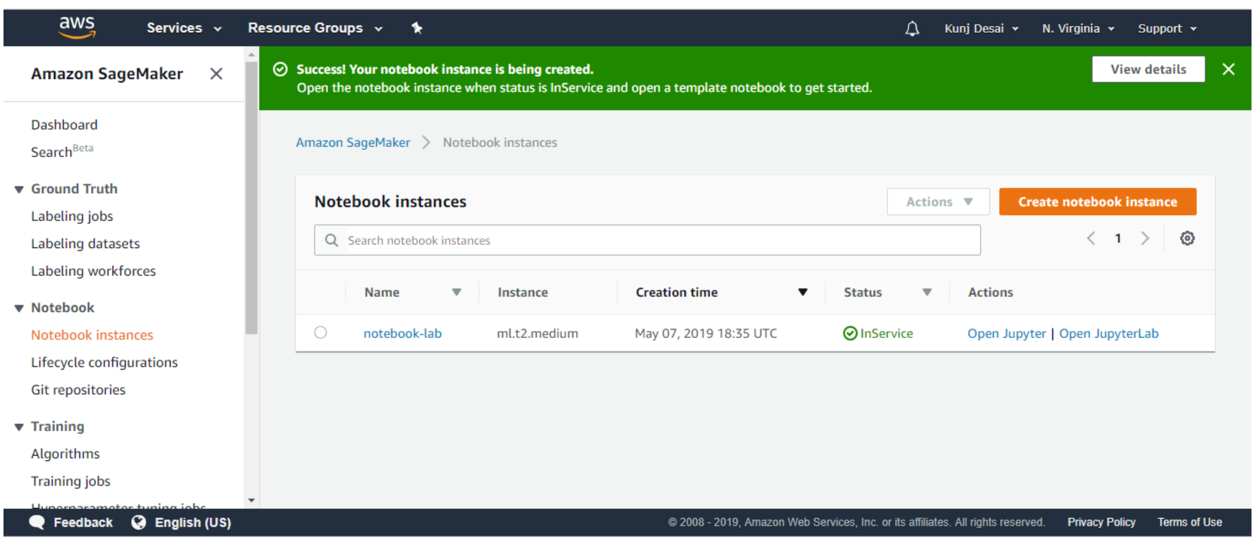
2. Then, I went to the Amazon SageMaker and opened "Notebook Instances" under the "Network" tab and created a new notebook instance.



3. I also created IAM role using the bucket I had.

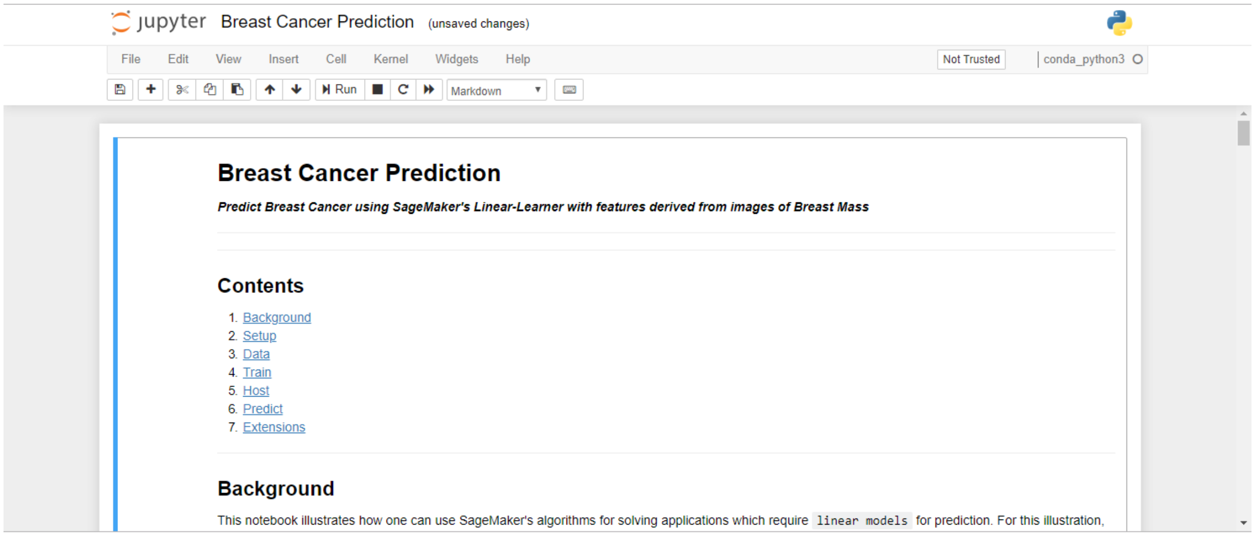


4. After the notebook instance is in service "Open Jupyter".



5. Under the "SageMaker Examples" tab, navigate to "Introduction to Applying Machine Learning" tab and find Breast Cancer Prediction.ipynb and click on use.

6. This will open a dialog asking for creating a copy in the home directory. Thus, clicking on “copy” create a copy of the same which will open jupyter page as follows:



7. Now edit the first cell with the name of the S3 bucket you created and run it. This cell is for setting up the S3 bucket where I will copy the data and model artifacts.

```
In [1]: import os
        import boto3
        import re
        from sagemaker import get_execution_role

        role = get_execution_role()

        bucket = 'bucketlab04'# enter your s3 bucket where you will copy data and model artifacts
        prefix = 'sagemaker/DEMO-breast-cancer-prediction' # place to upload training files within the bucket
```

Note: After every execution the In [ ] bracket will be filled by a number and while processing there will be an asterisk “*”.

8. Then execute the second cell which imports the python libraries needed.

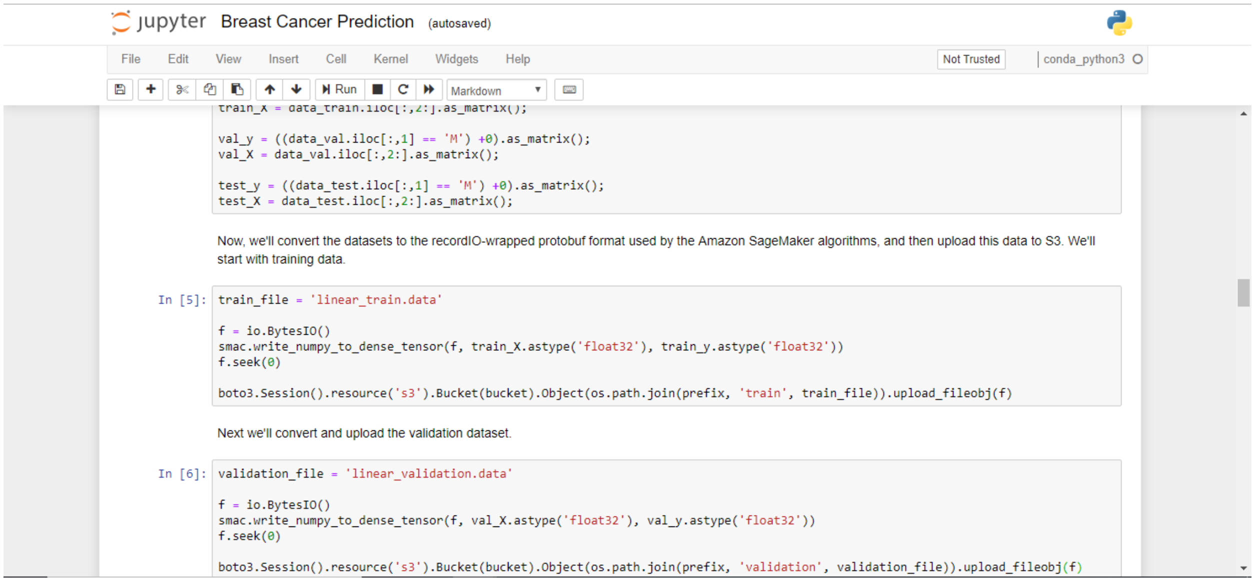
```
In [2]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import io
        import time
        import json
        import sagemaker.amazon.common as smac
```

9. Now execute the data cell which will give you the following observations and the relevant data to it:

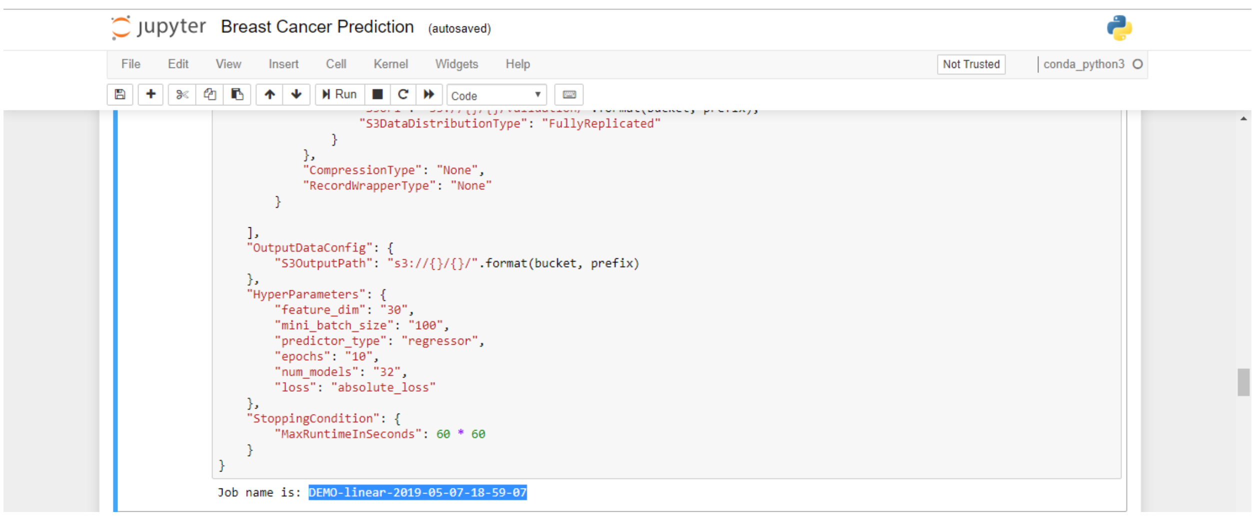
Key observations:

- Data has 569 observations and 32 columns.
- First field is 'id'.
- Second field, 'diagnosis', is an indicator of the actual diagnosis ('M' = Malignant; 'B' = Benign).
- There are 30 other numeric features available for prediction.

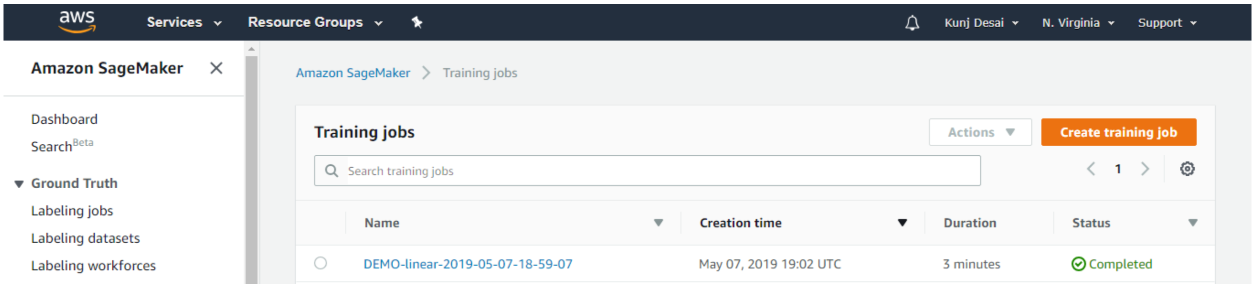
10. Now under Create Features and Labels, execute the cells which finally will convert and upload the validation dataset.



11. Now under the Train tab, execute three cells which will collectively create a job and will return its name. In my case, Job name is: DEMO-linear-2019-05-07-18-59-07



12. You can check the created job in the "Training job" under "Training" in the Amazon Sagemaker. When it shows the status completed, it means the job is been created.



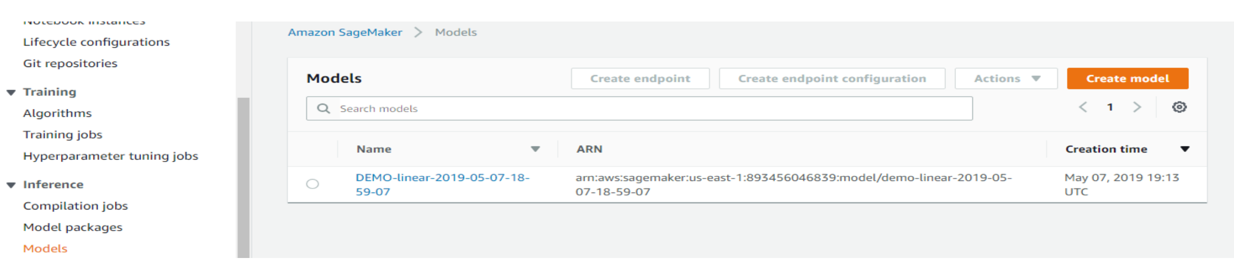
13. Now that we've trained the linear algorithm on our data, let's setup a model which can later be hosted. And then pointing it to the scoring container and creating the hosting model.

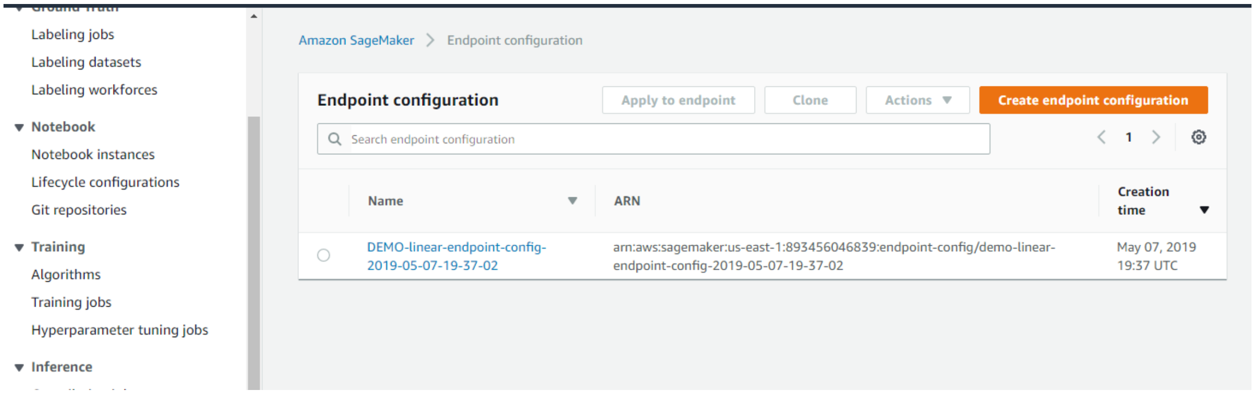
```
In [10]: linear_hosting_container = {
             'Image': container,
             'ModelDataUrl': sm.describe_training_job(TrainingJobName=linear_job)['ModelArtifacts']['S3ModelArtifacts']
         }

         create_model_response = sm.create_model(
             ModelName=linear_job,
             ExecutionRoleArn=role,
             PrimaryContainer=linear_hosting_container)

         print(create_model_response['ModelArn'])

         arn:aws:sagemaker:us-east-1:893456046839:model/demo-linear-2019-05-07-18-59-07
```

14. Once we've setup a model, we can configure what our hosting endpoints should be. Here we specify the EC2 instance type to use for hosting, initial number of instances and our hosting model name. Also, update the instance with "**m1.t2.medium**"

```
In [15]: linear_endpoint_config = 'DEMO-linear-endpoint-config-' + time.strftime("%Y-%m-%d-%H-%M-%S", time.gmtime())
         print(linear_endpoint_config)
         create_endpoint_config_response = sm.create_endpoint_config(
             EndpointConfigName=linear_endpoint_config,
             ProductionVariants=[{
                 'InstanceType': 'ml.t2.medium',
                 'InitialInstanceCount': 1,
                 'ModelName': linear_job,
                 'VariantName': 'AllTraffic'}])

         print("Endpoint Config Arn: " + create_endpoint_config_response['EndpointConfigArn'])

         DEMO-linear-endpoint-config-2019-05-07-19-37-02
         Endpoint Config Arn: arn:aws:sagemaker:us-east-1:893456046839:endpoint-config/demo-linear-endpoint-config-2019-05-07-19-37-02
```

Note: You can check the Models and the Endpoints tab in SageMaker to know the existence of models and endpoints created respectively.



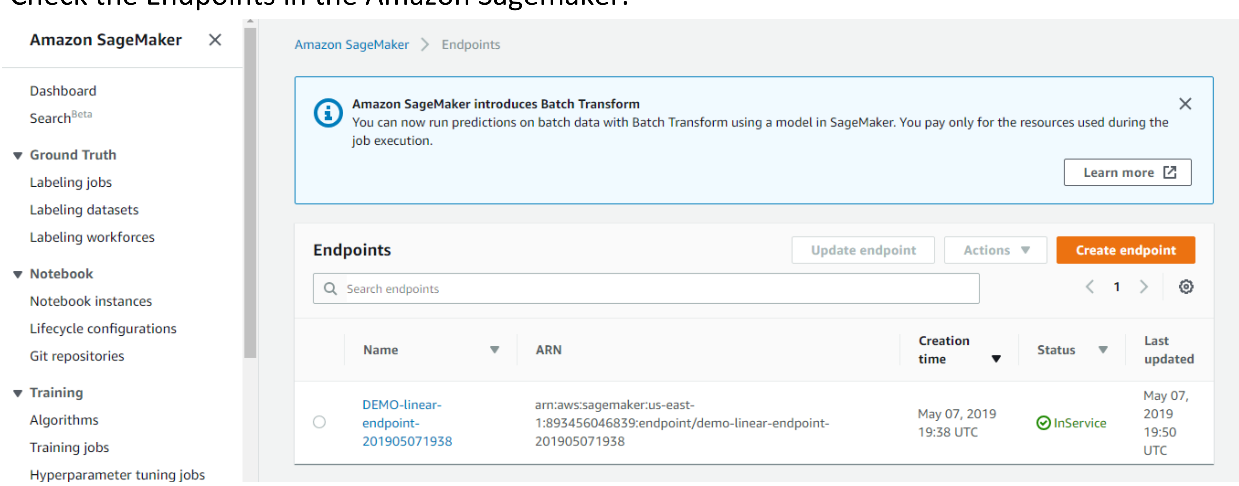


15. Now that we've specified how our endpoint should be configured, we can create/host them. This can be done in the background, but for now let's run a loop that updates us on the status of the endpoints so that we know when they are ready for use.

```
In [16]: %%time

linear_endpoint = 'DEMO-linear-endpoint-' + time.strftime("%Y%m%d%H%M", time.gmtime())
print(linear_endpoint)
create_endpoint_response = sm.create_endpoint(
    EndpointName=linear_endpoint,
    EndpointConfigName=linear_endpoint_config)
print(create_endpoint_response['EndpointArn'])

resp = sm.describe_endpoint(EndpointName=linear_endpoint)
status = resp['EndpointStatus']
print("Status: " + status)

sm.get_waiter('endpoint_in_service').wait(EndpointName=linear_endpoint)

resp = sm.describe_endpoint(EndpointName=linear_endpoint)
status = resp['EndpointStatus']
print("Arn: " + resp['EndpointArn'])
print("Status: " + status)

if status != 'InService':
    raise Exception('Endpoint creation did not succeed')
```

```
DEMO-linear-endpoint-201905071938
arn:aws:sagemaker:us-east-1:893456046839:endpoint/demo-linear-endpoint-201905071938
Status: Creating
Arn: arn:aws:sagemaker:us-east-1:893456046839:endpoint/demo-linear-endpoint-201905071938
Status: InService
CPU times: user 320 ms, sys: 2.01 ms, total: 322 ms
Wall time: 11min 32s
```

16. Check the Endpoints in the Amazon Sagemaker.



17. Now that we have our hosted endpoint, we can generate statistical predictions from it. Let's predict on our test dataset to understand how accurate our model is. Run the cells under the "Predict" tab.

jupyter Breast Cancer Prediction (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help — Not Trusted — conda_python3

Function to convert an array to a csv

In [17]:
```python
def np2csv(arr):
    csv = io.BytesIO()
    np.savetxt(csv, arr, delimiter=',', fmt='%g')
    return csv.getvalue().decode().rstrip()
```

Next, we'll invoke the endpoint to get predictions.

In [18]:
```python
runtime= boto3.client('runtime.sagemaker')

payload = np2csv(test_X)
response = runtime.invoke_endpoint(EndpointName=linear_endpoint,
                                   ContentType='text/csv',
                                   Body=payload)
result = json.loads(response['Body'].read().decode())
test_pred = np.array([r['score'] for r in result['predictions']])
```

18. Now we will compare linear learner based mean absolute prediction errors from a baseline prediction which uses majority class to predict every instance and predictive accuracy using a classification threshold of **0.5** for the predicted and compare against the majority class prediction from training data set.

Let's compare linear learner based mean absolute prediction errors from a baseline prediction which uses majority class to predict every instance.

In [19]:
```python
test_mae_linear = np.mean(np.abs(test_y - test_pred))
test_mae_baseline = np.mean(np.abs(test_y - np.median(train_y))) ## training median as baseline predictor

print("Test MAE Baseline :", round(test_mae_baseline, 3))
print("Test MAE Linear:", round(test_mae_linear,3))
```

```
Test MAE Baseline : 0.389
Test MAE Linear: 0.2
```

Let's compare predictive accuracy using a classification threshold of 0.5 for the predicted and compare against the majority class prediction from training data set

In [20]:
```python
test_pred_class = (test_pred > 0.5)+0;
test_pred_baseline = np.repeat(np.median(train_y), len(test_y))

prediction_accuracy = np.mean((test_y == test_pred_class))*100
baseline_accuracy = np.mean((test_y == test_pred_baseline))*100

print("Prediction Accuracy:", round(prediction_accuracy,1), "%")
print("Baseline Accuracy:", round(baseline_accuracy,1), "%")
```

```
Prediction Accuracy: 88.9 %
Baseline Accuracy: 61.1 %
```

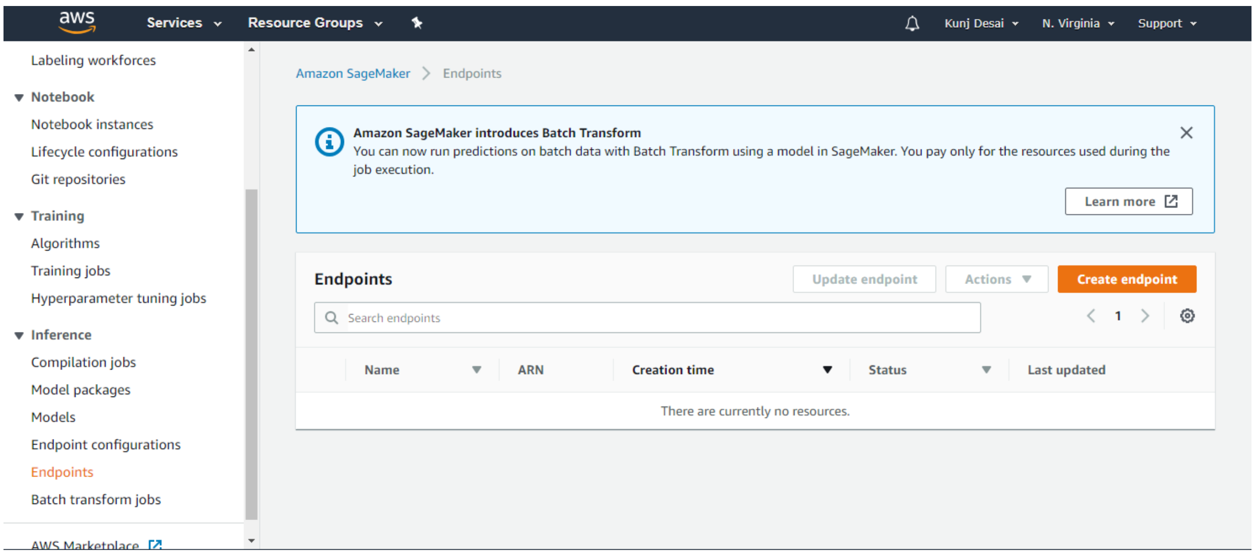
Thus, we completed the model with prediction accuracy of 88.9% and baseline accuracy of 61.1%.

19. Now let's clean the AWS by deleting the endpoint as we are done with the model. Run the last cell for the same.

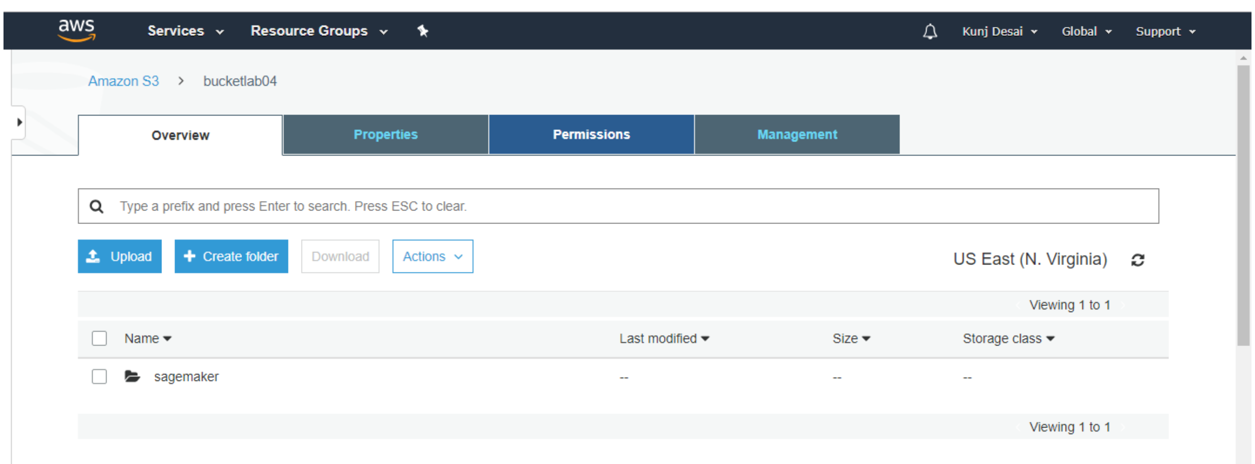
Run the cell below to delete endpoint once you are done.

```
In [21]: sm.delete_endpoint(EndpointName=linear_endpoint)

Out[21]: {'ResponseMetadata': {'RequestId': 'c1273aa8-b08d-47d9-bbb4-3d42b6f3216d',
           'HTTPStatusCode': 200,
           'HTTPHeaders': {'x-amzn-requestid': 'c1273aa8-b08d-47d9-bbb4-3d42b6f3216d',
            'content-type': 'application/x-amz-json-1.1',
            'content-length': '0',
            'date': 'Tue, 07 May 2019 20:11:24 GMT'},
           'RetryAttempts': 0}}
```

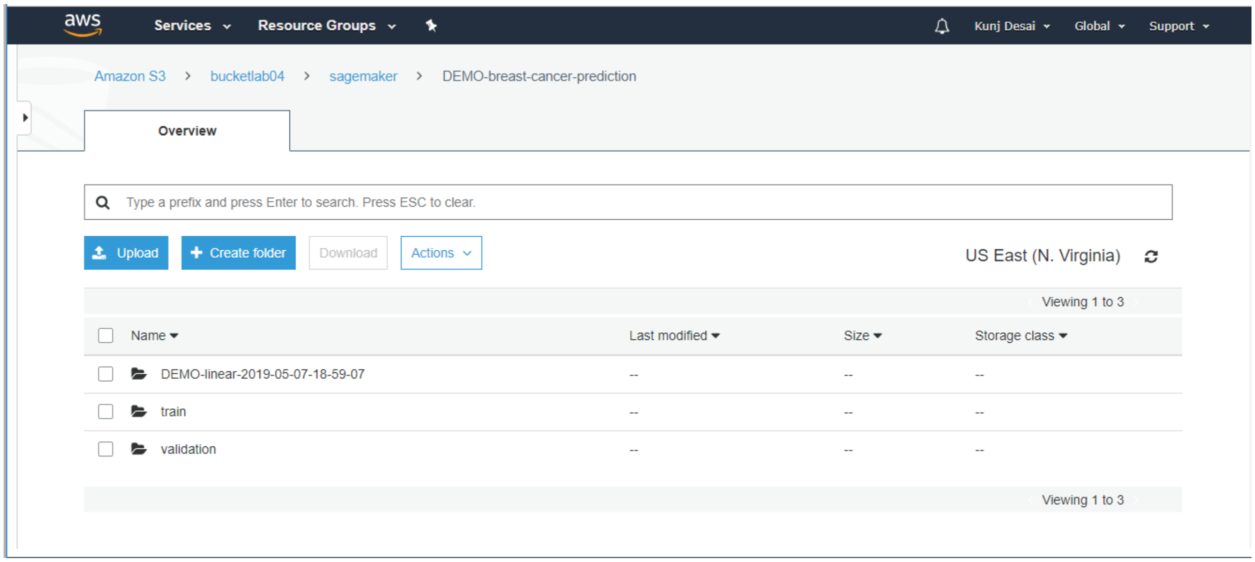
We can check there are no endpoints in the endpoints tab in Amazon SageMaker:



20. Now we can finally check the S3 bucket we created for the final output.



Note: Expanding the folder we can get relevant data information we need.



Conclusion: Thus, we completed the study of the model and got the relevant data output in the S3 bucket we created.