# Amazon Sagemaker는 처음이지? 누워서 머신러닝 학습부터 배포까지

안녕하세요!

Brandi LABS를 사랑해주시는 여러분들이 아 기다리고 기다리셨던 Brandi 개발자, 오- 연주 입니다.



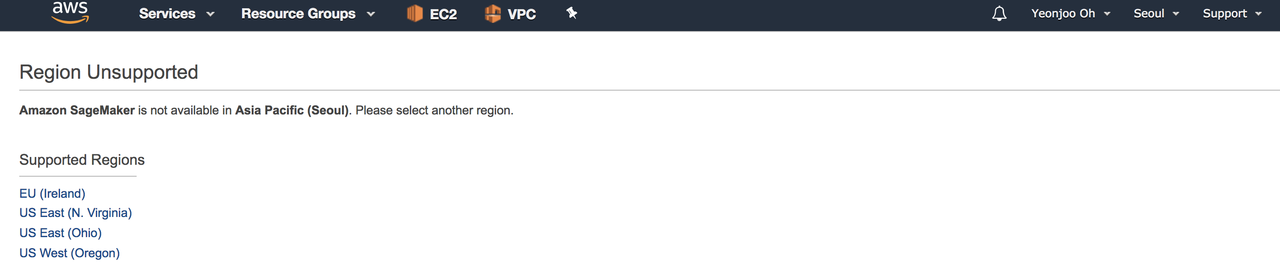
지난 4월, Brandi Back-end 개발자 분들과 코엑스에서 열렸던 **AWS Summit**(04.18 - 04.19)에 다녀왔습니다!

여러 세션을 듣는 와중에 우연히 AI machine learning 를 쉽게 도와주는 Cloud Machine learning Flatform인 Amazon SageMaker에 대해 들었습니다. 듣던 중 머닝러닝에서 학습을 시켜 그 데이터로 ‘Brandi 서비스와 연관지으면 어떨까’ 라는 생각을 하게 되었습니다.

그래서 오늘은 많은 분들의 관심사인 머신러닝 학습관련 Amazon Amazon SageMaker에 대해 글을 써볼까 합니다.



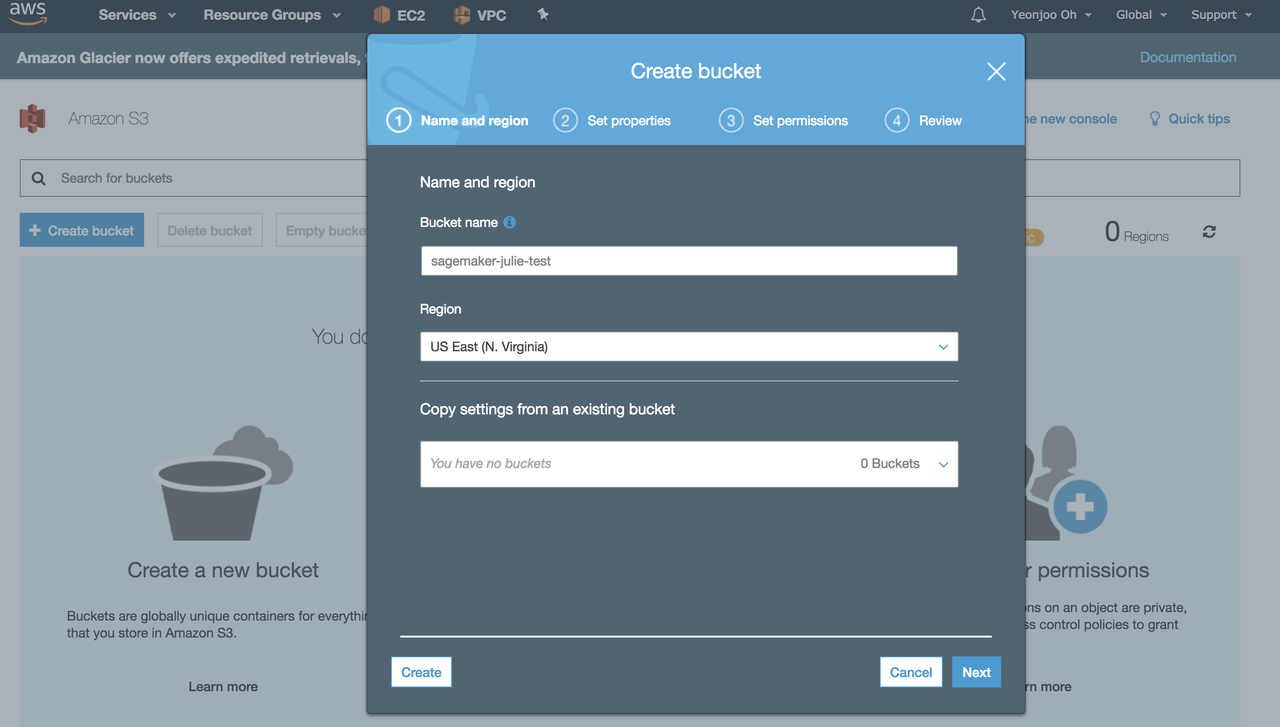
자, 퐈이팅 넘치게 신나게 sagemaker를 create해볼까요!!!! 했는데…



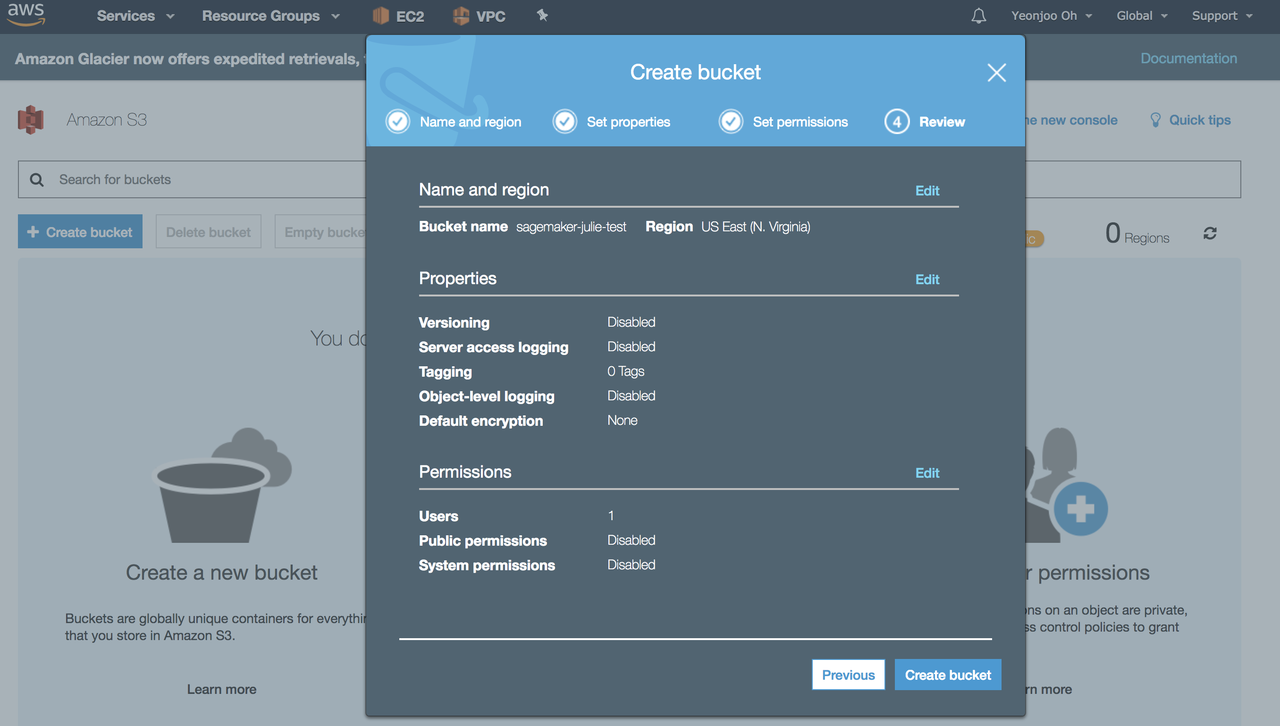
1. 현재 지원되는 리전은 아직 네 군데로, 저는 제일 있어보이는 미국 동부에 위치한 버지니아를 선택했어요 : )
2. 1] EU (Iceland) 2] US West (Oregon) 3] USEast (N. Virginia) 4] US East (Ohio)

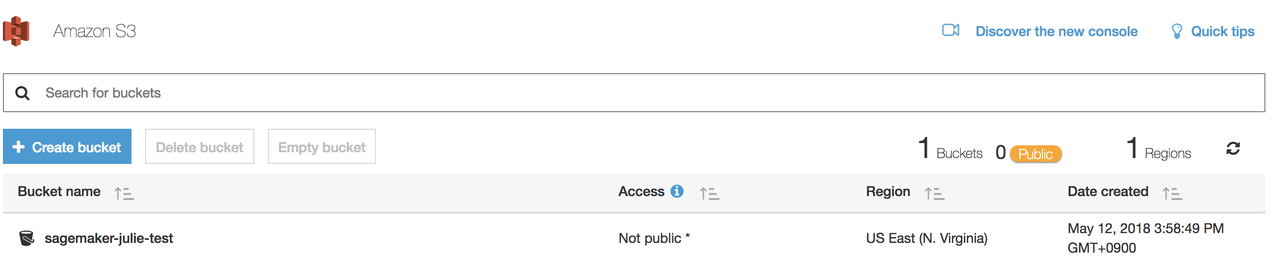
아 참! SageMaker를 create하기 전 학습할 데이터와 학습 모델을 저장할 S3 Bucket이 필요해요!

## 1. Default 값으로 S3를 만들어 주세요 : )

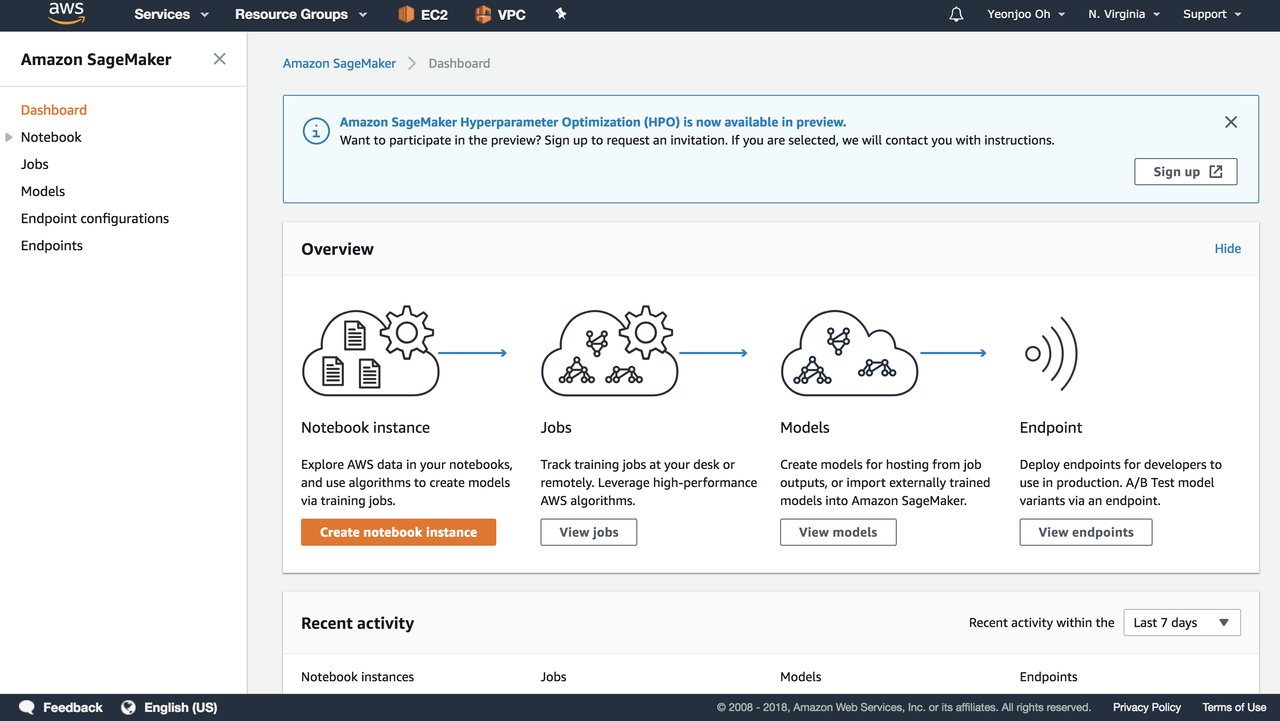


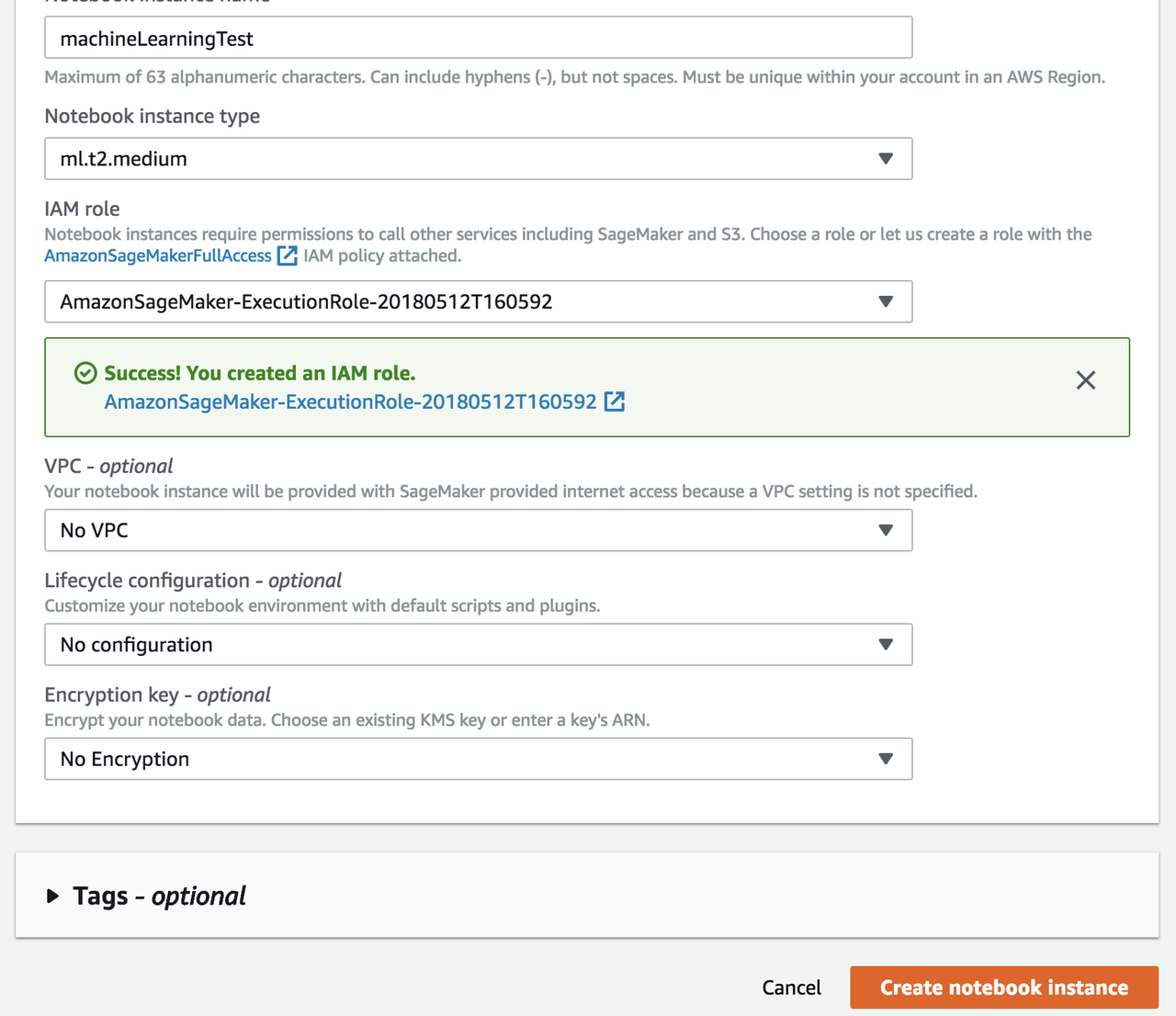
중요한 점은, bucket 이름이 “sagemaker-” 로 시작되어야 합니다. 그래야 나중에 notebook instance가 쉽게 어디에다가 데이터를 저장할지 알 수 있어요!





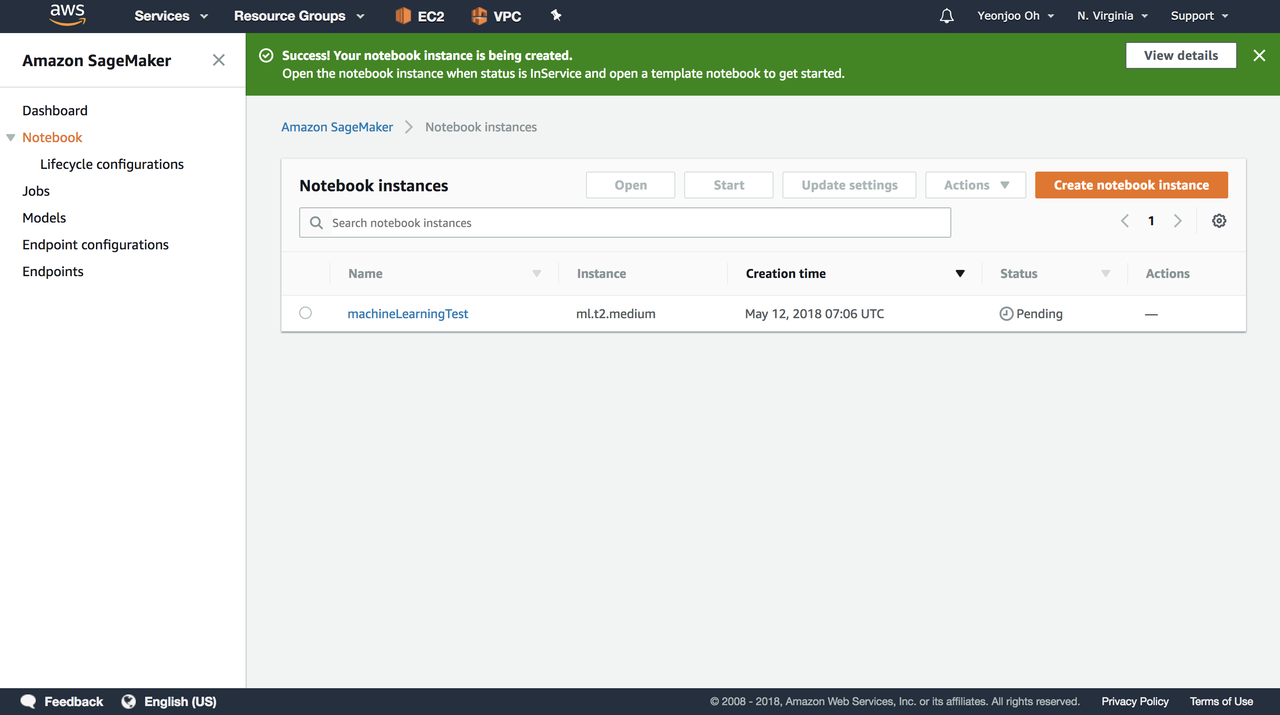
## 2.Create notebook instance 버튼을 눌러 SageMaker를 만들어 봅시다!





여러분들이 원하는 이름을 지어 줍니다. 저는 machineLearningTest 라고 지었어요.

IAM role 선택 하는 부분에서 None을 눌러 Default 값으로 sageMaker를 만들어 줍니다.



## 3. Pending이 끝나고 “open” action을 선택하면 Jupyter가 열릴거에요!



*Jupyter(Jupyter Notebook)*은 오픈 소스로 라이브 코드, 등식, 코드에 대한 시각화를 위하여 사용되며 또한 코드에 대한 문서화를 위해 description을 위한 텍스트 문서 (마크다운 등)을 지원하는 웹 어플리케이션 입니다.

앞으로 이 글에서 Jupyter Notebook을 통해 데이터를 학습하고, 그 데이터를 테스트를 할거에요!

***제가 진행한 전체 코드 스크립트(entire script)는 이 글의 맨 마지막 부분에 있으니 참고해 주세요, 찡긋 ㅇ\_<***

자, 이제 드디어 머신러닝 학습을 시켜볼 텐데요.

머신러닝 학습에 꼭 필요한 키워드 두 가지를 뽑아보았어요.

* + - **Dataset**: 정제된 데이터와 그 데이터에 대한 label을 정리해 놓은 데이터 모음
    - **Machine learning Algorithm**: 기계학습 알고리즘

저희는 MNIST 데이터셋을 k-means 알고리즘으로 학습시켜 볼 예정입니다.

**MNIST Dataset**

기계학습 알고리즘을 사용할 때 가장 기본적으로 테스트해보는 데이터셋으로 MNIST 데이터셋이 있습니다. 이것은 사람이 0부터 9까지 숫자 중 하나를 손글씨로 쓴 이미지 데이터와, 해당 이미지에 대한 레이블(0 - 9)이 6만 개 들어있는 학습 데이터셋입니다. 각 이미지는 가로와 세로가 각각 28 픽셀로서, 각 픽셀은 0부터 255 사이의 숫자가 들어있어요. 즉 하나의 이미지는 28 x 28 = 784개의 숫자로 이루어진 데이터입니다.

쉽게 말해 하나의 이미지를 나타나는 데이터의 array가 있느데 이 array의  length가 784라고 표현할 수 있겠네요.



**k-means**

지금 만든 SageMaker 학습 알고리즘은 AWS 튜토리얼에서 제시한 K-means를 사용할 예정입니다.

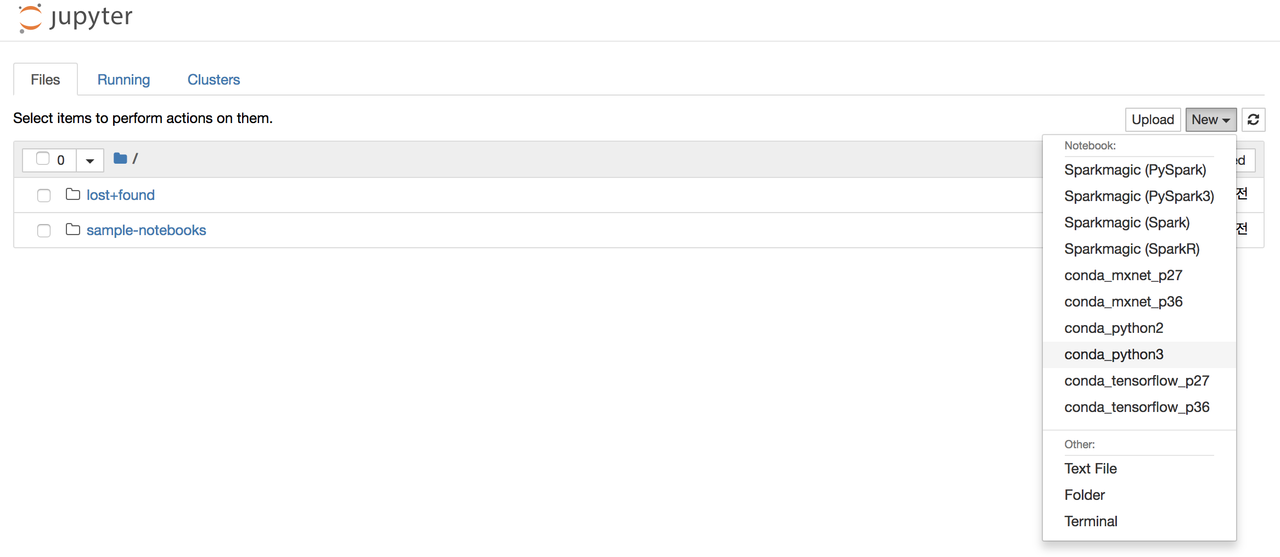
k-means는 label 없이, 즉 정답을 모르는 상태로 학습을 하는 비지도 학습 (unsupervised learning) 알고리즘 중 가장 쉽고 많이 쓰이는 알고리즘입니다. 정답을 모르니, ‘비슷한 애들끼리 뭉쳐봐’ 라고 하고, 알고리즘은 비슷한 친구들끼리 뭉쳐 놓게 됩니다. k-means에서 k는 ‘k개 덩어리로 뭉쳐주세요’라고 제시하는 숫자입니다. 저희는 0부터 9까지 비슷한 친구들끼리 모이게 하고 싶으니까 k=10을 쓸 겁니다.

지금 부터 해야 할 TODO!

1. MNIST 데이터셋을 다운로드받고, 우리가 학습시키기 좋도록 정제하기(preprocessing)
2. Amazon SageMaker를 통하여 데이터 학습시키기(training job)
3. Amazon SageMaker를 통하여 학습된 데이터를 배포하기(Deploy the model)
4. 배포된 모델에 요청을 보내 테스트 데이터에 대한 예측값을 받아오기(inference)

## 4. Jupyter 노트북 인스턴스 생성하기

Jupyter에 New Notebook(conda\_python3)를 선택해 새로운 노트북을 생성해 주세요.



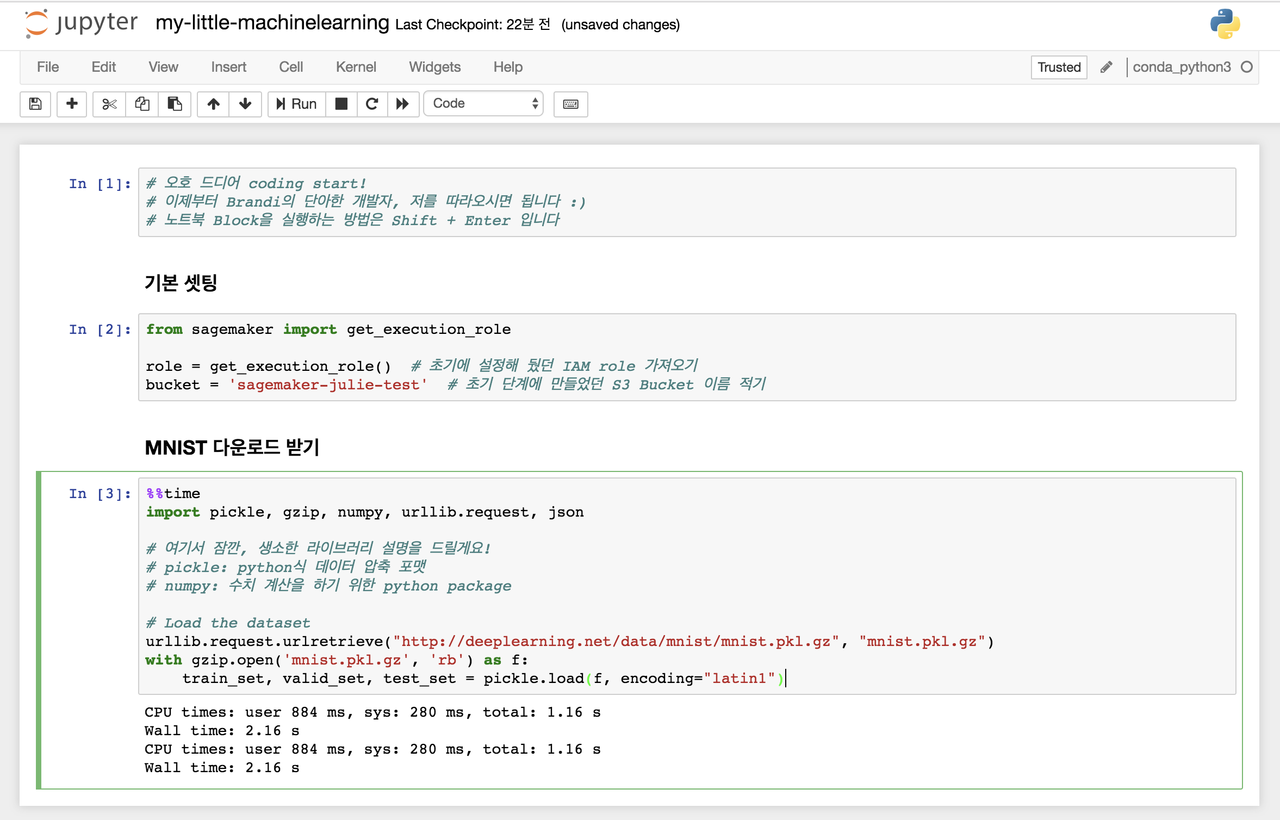
## 5. 학습시키기 위한 기본 셋팅

드디어 코딩 시작입니다! (의욕활활)

초기 설정해 주었던 IAM role, S3 Bucket, MNIST 다운로드, 다운받은 데이터 확인 등을 해주세요.

글보다 코드를 통해 주석을 보는게 가독성이 좋을 것 같아요.

아래 노트북을 통해 마크다운, 주석처리를 통해 description을 해놓았으니 참고해 주세요.



외부에서 MNIST 다운로드가 쉽도록 해준 url을 가지고 MNIST를 다운받는데 성공했습니다. MNIST 데이터셋 내용물 중 하나를 jupyter notebook에 그려서 제대로 다운받았나 show\_digit() 함수를 작성해 확인해볼까요?



## 6. 머신러닝 학습하기

이 세션에서는 기계학습 알고리즘 설정, 학습할 데이터 경로를 지정할 거에요.

그 후 MNIST 학습 데이터를 S3 버킷에 옮겨 저장하게 되죠.



kmeans.fit() 함수를 호출해 직접 학습을 시켜볼거에요!

학습 과정은 상당히 오래 걸린다고 써져 있었는데 다행히도 4분만에 학습이 끝났네요?



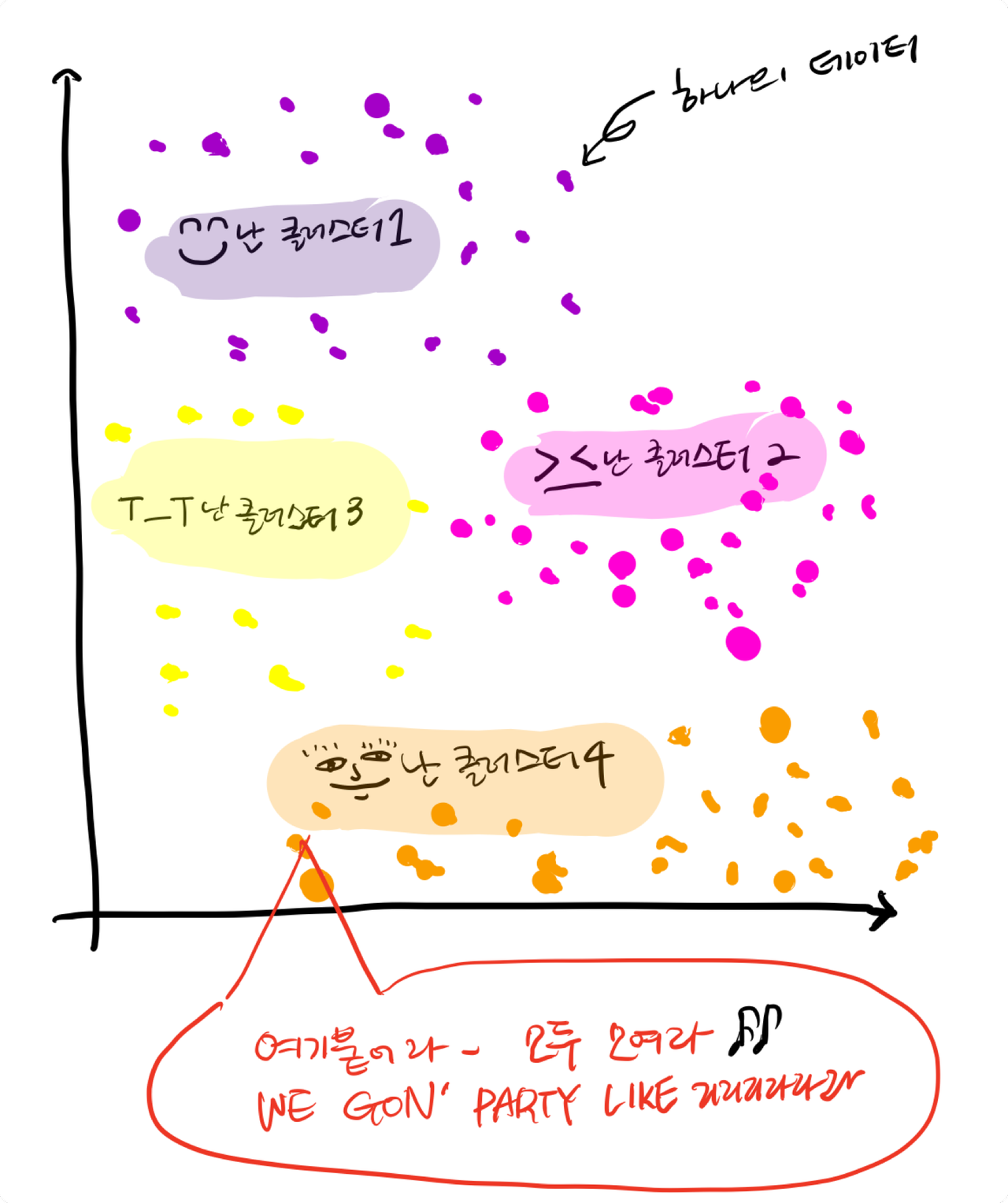
여기서 잠깐!

자 여기서 k = 10에 대해서 조금 더 알아보도록 할게요

cluster란 한 지점에 점을 찍어 놓고 데이터 분석을 한 뒤 비슷한 데이터들의 군집을 만들어 주는 것입니다.

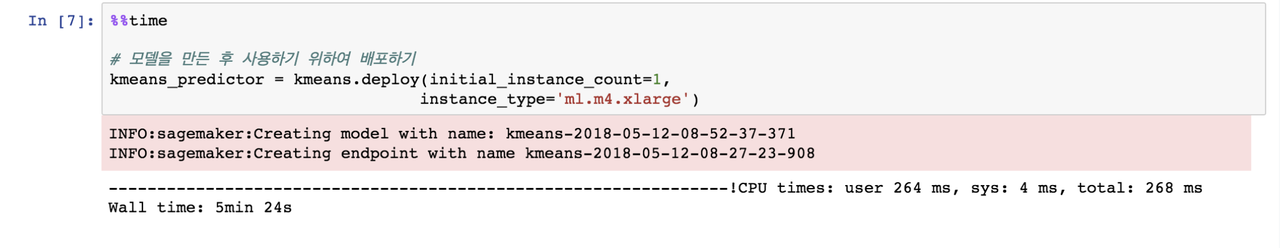
k-means가 진행되면서 각 cluster의 중심이 더욱 서로가 잘 뭉치는 방향으로 이동하게 됩니다.

더 와닿는 설명을 위하여 직접 그려보았어요(부끄)



## 7. 학습된 모델을 배포하기

학습을 시킨 뒤에는 이것을 테스트해보거나 사용할 수 있어야겠죠? 학습된 모델을 배포해 주세요.



## 8. 배포된 모델 테스트 진행하기

배포된 모델에 valid\_set 데이터로 검증 데이터 진행합니다.



.predict() 함수를 호출하면 새로운 이미지가 어떤 cluster에 속했는지 예측 결과를 알려줍니다. 가장 가까운 cluster가 0번이라고 예측 결과를 반환했네요. 또한 cluster 중심과의 거리는 5.85라고 알려줍니다. 여기서 중요한 점은 cluster 번호와 실제 숫자는 일치하지 않는다는 것입니다. 알고리즘은 임의로 cluster 중심에 번호를 매기는데, 꼭 0번 클러스터가 숫자 ’0’을 뭉쳐놓은 것은 아니에요!

## 9. 데이터 예측해보기

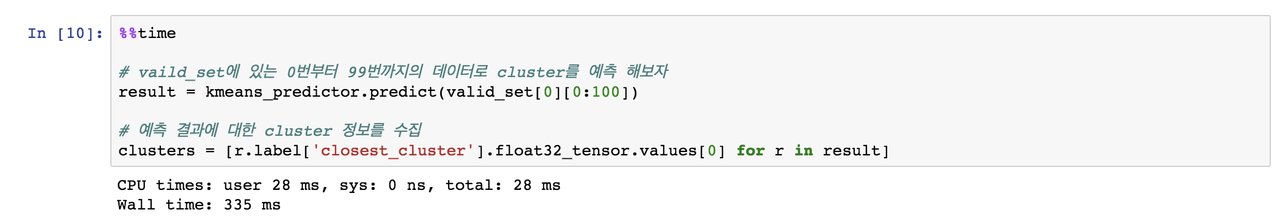
이제 조금 더 많은 데이터를 예측해볼까요?

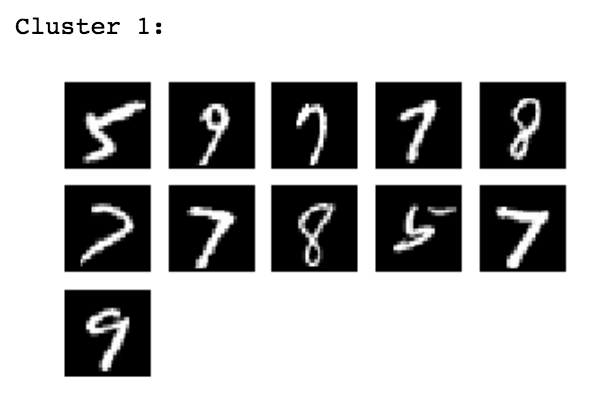
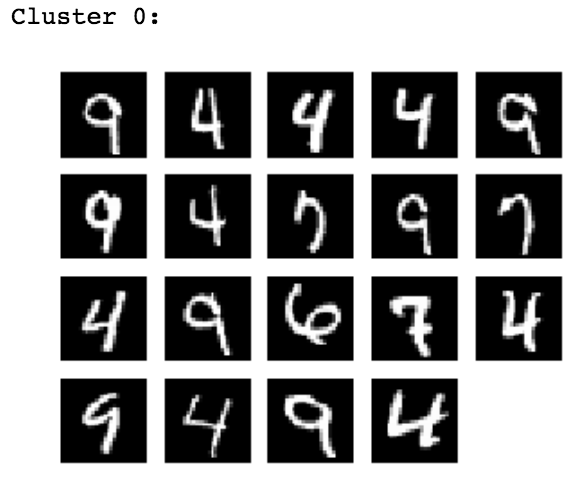
valid set에 있는 100개 데이터를 예측해봅시다!

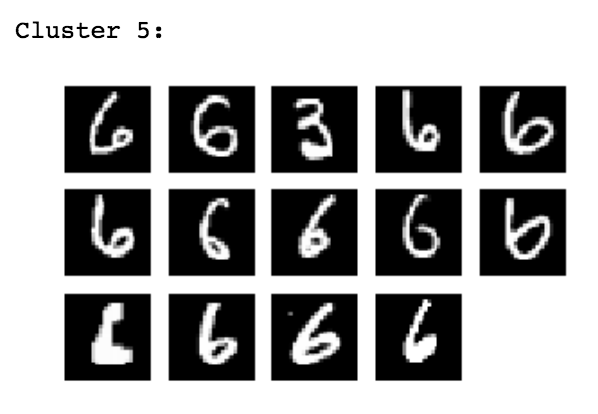
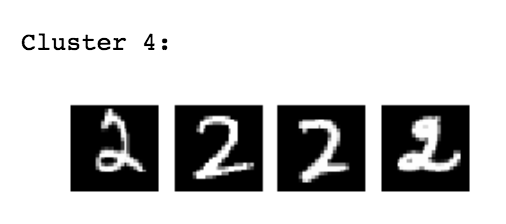
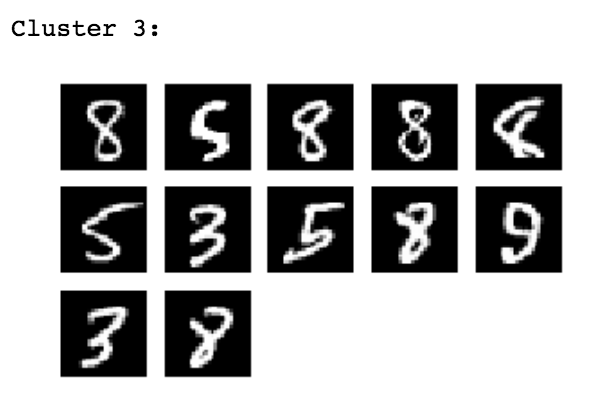
각 cluster에 가까운 데이터들이 쭉 뽑혔습니다!

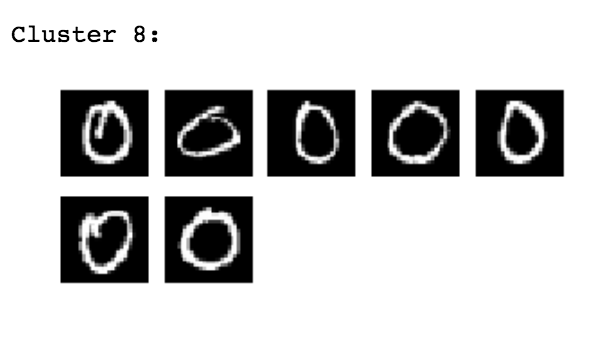
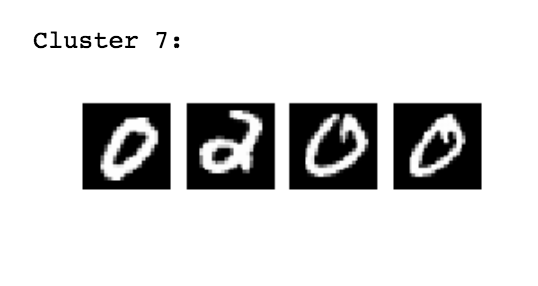
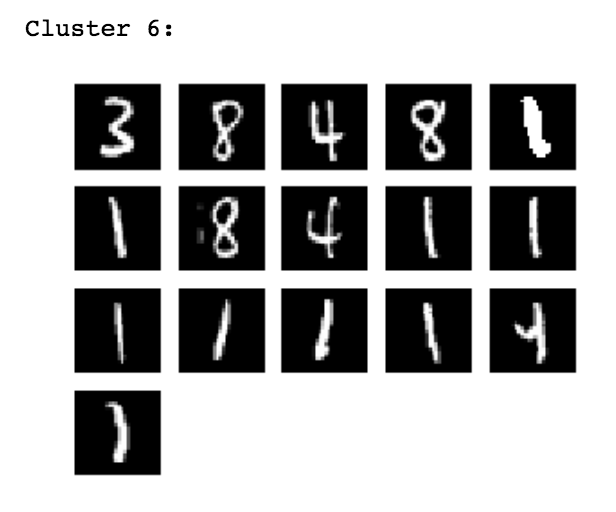
정확하지는 않지만 비슷한 숫자 모양들이 서로 군집되어 보여주고 있습니다.

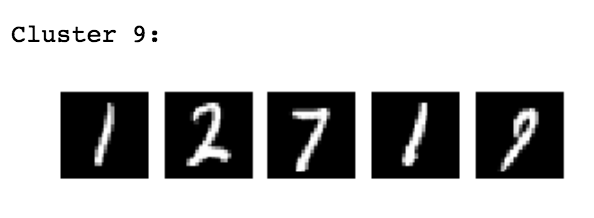
0과 2같은 숫자들은 잘 표현되지만, 알고리즘이 9랑 4를 헷갈리거나 5와 3을 헷갈리는 듯 하네요.











[MISSING IMAGE: , ]

# FASHION MNIST로 SageMaker 머신러닝 학습 및 예측해보기

자, 이제 몸도 풀어보았습니다!

제가 하고 싶었던 패션관련 머신러닝 학습 및 예측하는 것을 진행해 볼게요.

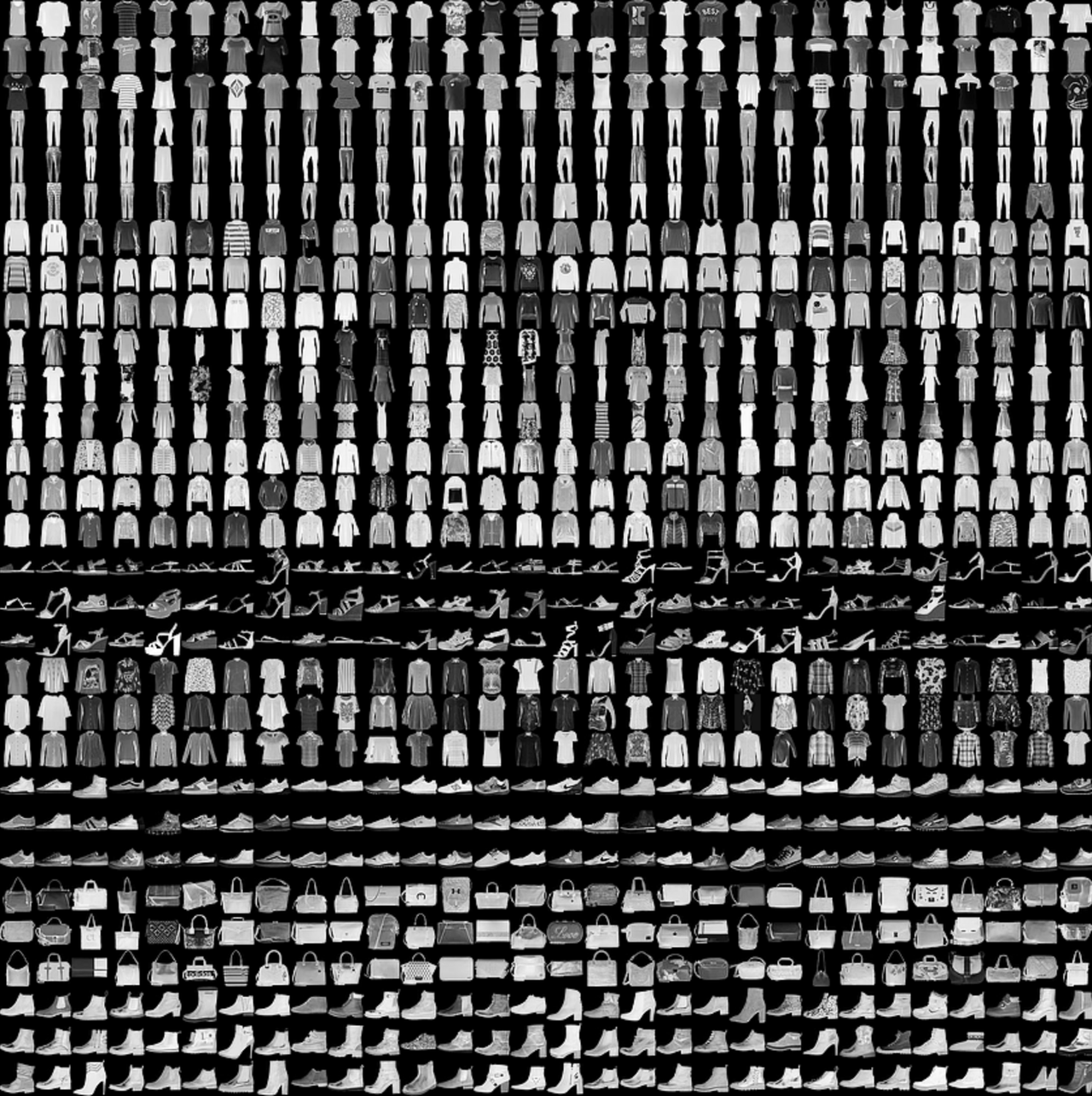
마침 옷 그림을 가지고 MNIST와 매우 비슷한 데이터를 만들어 놓은 fashion-MNIST라는 데이터셋을 발견했어요!

## 1. 패션 관련 Mnist 다운로드 받기

패션 MNIST 데이터셋을 우선 다운받아 볼게요! 다운로드는 [여기](https://github.com/zalandoresearch/fashion-mnist) 에서 받을 수 있습니다.

총 네 개 파일을 다운로드 받으면 됩니다.

* train-images-idx3-ubyte.gz : train set 이미지
* train-labels-idx1-ubyte.gz : train set 레이블
* t10k-images-idx3-ubyte.gz : test set 이미지
* t10k-labels-idx1-ubyte.gz : test set 레이블



다운로드 받은 패션 Mnist의 label은 아래와 같이 되어 있습니다. 숫자 0부터 9 대신에 각 이미지가 어떤 이미지인지 텍스트로 표현되어 있어요.

|  |  |
| --- | --- |
| **Label** | **Description** |
| 0 | T-shirt/top |
| 1 | Trouser |
| 2 | Pullover |
| 3 | Dress |
| 4 | Coat |
| 5 | Sandal |
| 6 | Shirt |
| 7 | Sneaker |
| 8 | Bag |
| 9 | Ankle boot |

## 2. Fashion-MNIST 데이터셋을 이전에 사용했던 mnist.pkl.gz와 같은 형태로 변환해주는 스크립트 작성해주기

위에서 연습할 때는 mnist.pkl.gz 한개 파일만 사용했는데요!?!

그래서 다운로드받은 네개 파일을 똑같은 형식의 파일 하나로 만들어주는 파이썬 스크립트를 작성하여 fashion-mnist.pkl.gz 파일로 만들어주었어요.

import gzip

import pickle

import numpy as np

# MNIST 데이터셋은 train, test 셋이 각각 image, label로 나누어 저장되어있는 4개의 파일로 구성

test\_image\_path = 't10k-images-idx3-ubyte.gz'

test\_label\_path = 't10k-labels-idx1-ubyte.gz'

train\_label\_path = 'train-labels-idx1-ubyte.gz'

train\_image\_path = 'train-images-idx3-ubyte.gz'

out\_file\_name = 'fashion-mnist.pkl.gz'

# train label / images 추출

with gzip.open(train\_label\_path, 'rb') as train\_label\_f:

    train\_label = np.frombuffer(

            train\_label\_f.read(), dtype=np.uint8, offset=8).astype(np.int64)

with gzip.open(train\_image\_path, 'rb') as train\_image\_f:

    train\_imgs = np.frombuffer(

            train\_image\_f.read(), dtype=np.uint8, offset=16).reshape(-1, 784).astype(np.float32)

# test label / images 추출

with gzip.open(test\_label\_path, 'rb') as test\_label\_f:

    test\_label = np.frombuffer(test\_label\_f.read(), dtype=np.uint8, offset=8).astype(np.int64)

with gzip.open(test\_image\_path, 'rb') as test\_image\_f:

    test\_imgs = np.frombuffer(

            test\_image\_f.read(), dtype=np.uint8, offset=16).reshape(-1, 784).astype(np.float32)

# 기존 60000개 training set에서 50000개는 train set으로 사용하고, 10000개는 valid set으로 활용

train\_label, valid\_label = train\_label[:50000], train\_label[50000:]

train\_imgs, valid\_imgs = train\_imgs[:50000], train\_imgs[50000:]

# train set, validati on set, test set을 튜플 자료형으로 저장

out\_data = ((train\_imgs, train\_label),

            (valid\_imgs, valid\_label),

            (test\_imgs, test\_label))

# pickle file로 dataset 데이터 포맷 맞춰주기

with gzip.open(out\_file\_name, 'wb') as out\_f:

    pickle.dump(out\_data, out\_f)

이렇게 나온 결과물인 fashion-mnist.pkl.gz를 Jupyter Notebook이 있는 경로에 업로드합니다.



## 3. 머신러닝 학습하기

아까 사용했던 활용했던 숫자 MNIST 스크립트를 그대로 사용할 거에요!

 show\_digit()을 이름만 바꾼 show\_fashion()으로 데이터를 살펴보니 드레스가 있네요.

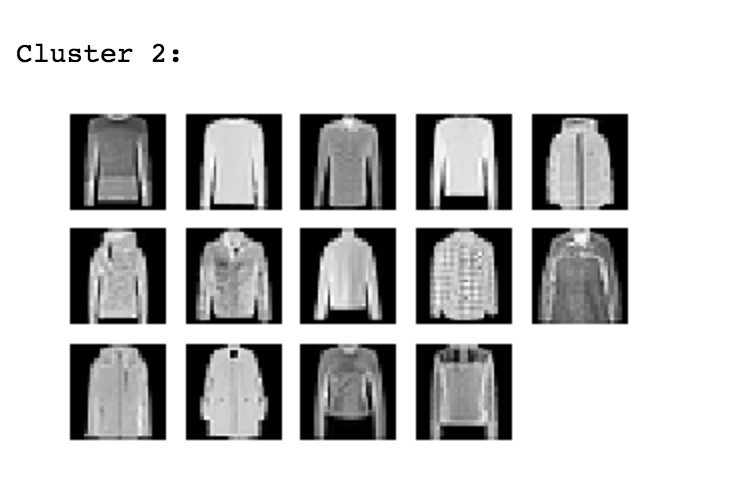
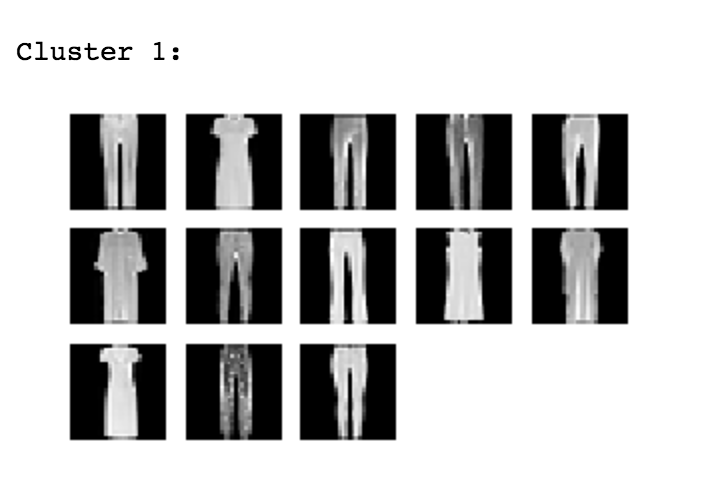


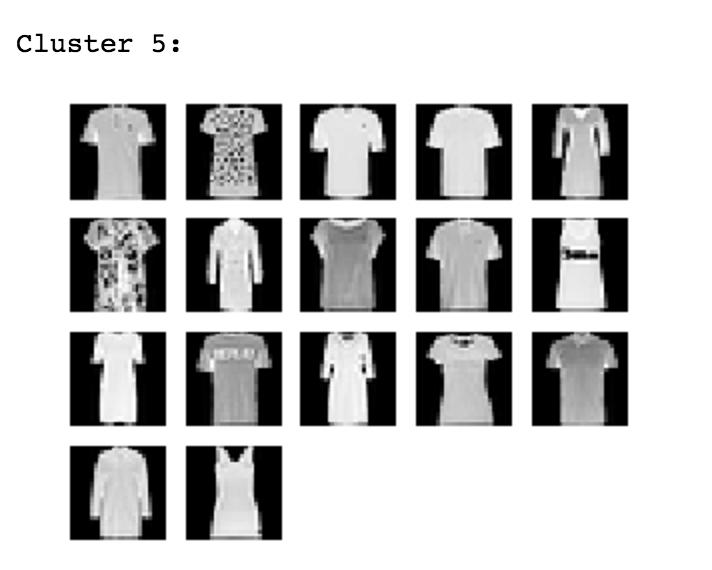
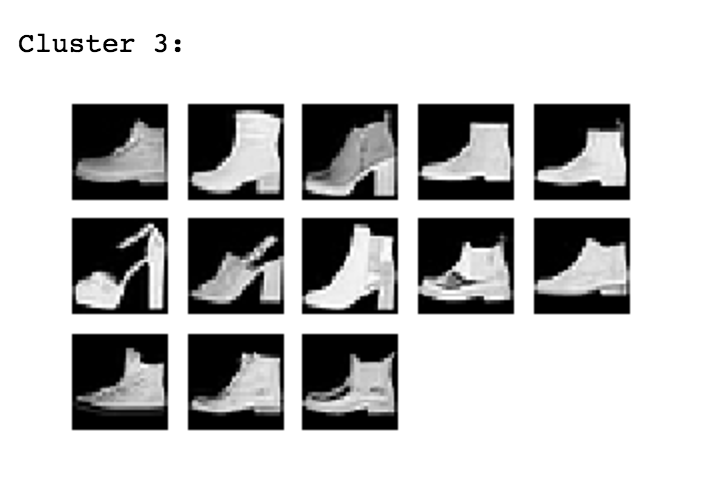
아까 전에 했던 숫자 MNIST와 똑같은 과정을 SageMaker를 통해,

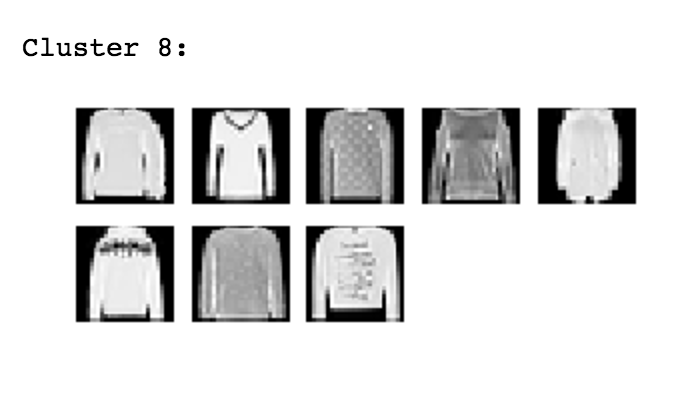
학습 → 테스트 → 예측해보니 아래와 같은 예측결과를 얻을 수 있어요!

신발은 신발끼리, 바지는 바지끼리, 가방은 가방끼리 뭉쳐있는게 너무나 신기하네요.

아까 했던 숫자보다 더 학습이 잘 된 것 같은건 기분 탓인가요…?





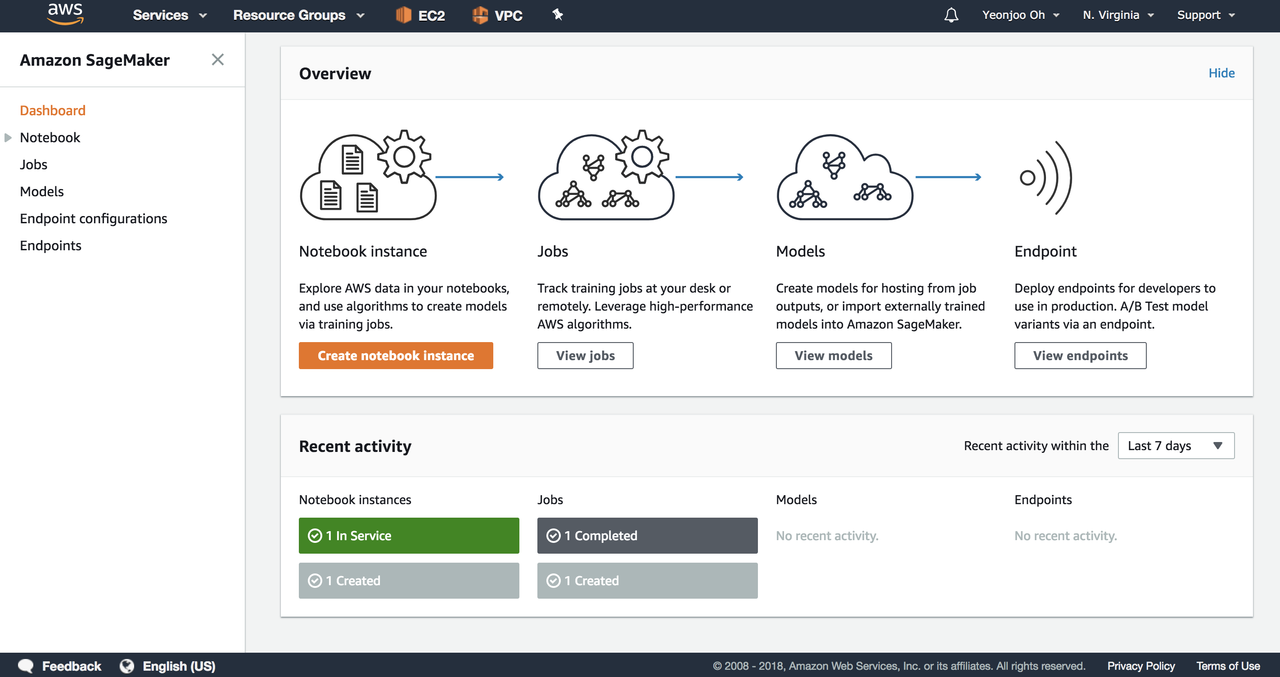




이렇게 AWS SageMaker를 통하여 머신러닝 학습부터 배포 그리고 예측까지 보다 더 쉽게 다가갈 수 있었습니다.

머신러닝이라고 겁내지 말고 유저들에게 더 좋은 서비스 제공할 수 있을 것 같네요.

지금까지 Brandi 개발 2팀, 단아한 개발자 오연ㅈ….



**잠시만요!! 중.. 중요한 것을 빼놓을 뻔 했네요.**

저처럼 이렇게 테스트를 하시고 그냥 지나치시지 말고 자동으로 결제되어 출금되는 뼈아픈 경험을 하지 마시고

꼭 꼭 이용하셨던 서비스들을 stop 시키시거나 terminate 시켜주세요.(Clean-up단계)

자세한 내용은 아래 URL을 꾸욱 [이곳을 눌러주세요.](https://docs.aws.amazon.com/sagemaker/latest/dg/ex1-cleanup.html)

지금까지 Brandi 개발 2팀, 단아한 개발자 오연주였습니다!

# entire script (숫자 Mnist)

# 오호 드디어 coding start!

# 이제부터 Brandi의 단아한 개발자, 저를 따라오시면 됩니다 :)

# 노트북 Block을 실행하는 방법은 Shift + Enter 입니다

from sagemaker import get\_execution\_role

role = get\_execution\_role()  # 초기에 설정해 뒀던 IAM role 가져오기

bucket = 'sagemaker-julie-test'  # 초기 단계에 만들었던 S3 Bucket 이름 적기

%%time

import pickle, gzip, numpy, urllib.request, json

# 여기서 잠깐, 생소한 라이브러리 설명을 드릴게요!

# pickle: python식 데이터 압축 포맷

# numpy: 수치 계산을 하기 위한 python package

# Load the dataset

urllib.request.urlretrieve("http://deeplearning.net/data/mnist/mnist.pkl.gz", "mnist.pkl.gz")

with gzip.open('mnist.pkl.gz', 'rb') as f:

    train\_set, valid\_set, test\_set = pickle.load(f, encoding="latin1")

# matplotlib로 그리는 그림이 jupyter 노트북에 바로 보여줄 수 있도록 설정

%matplotlib inline

import matplotlib.pyplot as plt  # 도표나 그림을 그릴 수 있게 해주는 라이브러리

plt.rcParams["figure.figsize"] = (2, 10)  # 그림의 크기 지정

def show\_digit(img, caption='', subplot=None):

    if subplot is None:

        \_,(subplot) = plt.subplots(1,1)

    imgr = img.reshape((28, 28))

    subplot.axis('off')

    subplot.imshow(imgr, cmap='gray')

    plt.title(caption)

# train\_set의 그림과[0] 데이터 이름[1]을 예시로 보여준다

show\_digit(train\_set[0][30], 'This is a {}'.format(train\_set[1][30]))

# 학습을 하기 위해 학습 알고리즘 및 데이터 경로 설정!

from sagemaker import KMeans

data\_location = 's3://{}/kmeans\_highlevel\_example/data'.format(bucket)

output\_location = 's3://{}/kmeans\_example/output'.format(bucket)

print('training data will be uploaded to: {}'.format(data\_location))

print('training artifacts will be uploaded to: {}'.format(output\_location))

kmeans = KMeans(role=role,

                train\_instance\_count=2,  # 장비 2대를 사용하여 학습하겠어요!

                train\_instance\_type='ml.c4.8xlarge',

                output\_path=output\_location,

                k=10,  # 아래 그림을 참고해 주세요!

                data\_location=data\_location)

%%time

# 학습 시작!

kmeans.fit(kmeans.record\_set(train\_set[0]))

%%time

# 모델을 만든 후 사용하기 위하여 배포하기

kmeans\_predictor = kmeans.deploy(initial\_instance\_count=1,

                                instance\_type='ml.m4.xlarge')

# valid\_set에 30번째 sample을 테스트 해보기

result = kmeans\_predictor.predict(valid\_set[0][30:31])

print(result)

%%time

# vaild\_set에 있는 0번부터 99번까지의 데이터로 cluster를 예측 해보자

result = kmeans\_predictor.predict(valid\_set[0][0:100])

# 예측 결과에 대한 cluster 정보를 수집

clusters = [r.label['closest\_cluster'].float32\_tensor.values[0] for r in result]

# 각 cluster별 예측된 이미지 출력

for cluster in range(10):

    print('\n\n\nCluster {}:'.format(int(cluster)))

    digits = [ img for l, img in zip(clusters, valid\_set[0]) if int(l) == cluster ]

    height = ((len(digits)-1)//5)+1

    width = 5

    plt.rcParams["figure.figsize"] = (width,height)

    \_, subplots = plt.subplots(height, width)

    subplots = numpy.ndarray.flatten(subplots)

    for subplot, image in zip(subplots, digits):

        show\_digit(image, subplot=subplot)

    for subplot in subplots[len(digits):]:

        subplot.axis('off')

    plt.show()

출처:

<https://docs.aws.amazon.com/sagemaker/latest/dg/gs.html>

<https://lh3.googleusercontent.com/3TTmLuiblRSNDiuMnNorH4z0O3ndD0cDvUg1KYTdKQYm4NsQAeSpXfj7-Tge0cjnukt05Q=s121>

<https://codeonweb.com/entry/12045839-0aa9-4bad-8c7e-336b89401e10>

<https://github.com/zalandoresearch/fashion-mnist>