



# Google colaboratory (Cloud computing)

Colaboratory에 오신 것을 환영합니다

파일 수정 보기 삽입 런타임 도구 도움말

Colaboratory란?

줄여서 'Colab'이라고도 하는 Colaboratory를 사용하면 브라우저에서 Python을 작성하고 실행할 수 있습니다. Colab은 다음과 같은 이점을 자랑합니다.

- 구성이 필요하지 않음
- GPU 무료 액세스
- 간편한 공유

학생이든, 데이터 과학자든, AI 연구원이든 Colab으로 업무를 더욱 간편하게 처리할 수 있습니다. [Colab 소개 영상](#)에서 자세한 내용을 확인하거나 아래에서 시작해 보세요.

시작하기

지금 읽고 계신 문서는 정적 웹페이지가 아니라 코드를 작성하고 실행할 수 있는 대화형 환경인 Colab 메모장입니다. 예를 들어 다음은 값을 계산하여 변수로 저장하고 결과를 출력하는 간단한 Python 스크립트가 포함된 코드 셀입니다.

```
seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day
```

86400

위 셀의 코드를 실행하려면 셀을 클릭하여 선택한 후 코드 왼쪽의 실행 버튼을 누르거나 단축키 'Command/Ctrl+Enter'를 사용하세요. 셀을 클릭하면 코드 수정을 바로 시작할 수 있습니다.

특정 셀에서 정의한 변수를 나중에 다른 셀에서 사용할 수 있습니다.

```
[ ] seconds_in_a_week = 7 * seconds_in_a_day
seconds_in_a_week
```

604800

웹 사이트 접속!

<http://colab.research.google.com>



별도의 복잡한 설치 작업이  
없이 간단한 셋팅으로 GPU  
등을 사용할 수 있다!

This notebook is open with private outputs. Outputs will not be saved. You can disable this in [Notebook settings](#).

ds-ai-test.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

Table of contents X

- Test for DS/AI class
  - Section

+ Code + Text

RAM Disk Editing ^

### Test for DS/AI class

Server spec!

```
1 from tensorflow.python.client import device_lib
2 device_lib.list_local_devices()
```

```
[name: "/device:CPU:0"
device_type: "CPU"
memory_limit: 268435456
locality {
}
incarnation: 17670787319713240863, name: "/device:GPU:0"
device_type: "GPU"
memory_limit: 14638920512
locality {
  bus_id: 1
  links {
  }
}
incarnation: 9972398538379896466
physical_device_desc: "device: 0, name: Tesla T4, pci bus id: 0000:00:04.0, compute capability: 7.5"]
```

2



무엇을 찾으시나요?

검색

로그인

장바구니

KOR

제품 솔루션 서비스 지원 추천할인

대한민국 > 업무용 > 액세서리 > 전원 및 누전 방지 > 그래픽 및 비디오 카드 > Dell 16GB NVIDIA Tesla T4 GPU 그래픽 카드

## Dell 16GB NVIDIA Tesla T4 GPU 그래픽 카드

NVIDIA Tesla GPU 가속기로 요구 사양이 높은 HPC, 하이퍼스케일, 엔터프라이즈 데이터 센터 작업을 가속화하세요.

이제 에너지 탐사에서 머신 러닝에 이르기까지, 다양한 분야의 과학자들이 CPU를 사용하는 것보다 빠른 속도로 페타바이트 단위의 데이터를 조사할 수 있습니다. 또한 Tesla 가속기는 대형 시뮬레이션을 이전보다 빠른 속도로 실행할 수 있는 성능을 제공합니다. VDI를 배포한 기업에게 있어 Tesla 가속기는 모든 사용자에게 어디서 ... [더 보기](#)

**3,969,636 원** 7,783,600 원

할인 **3,813,964 원**

10% 부가세 포함

장바구니에 담기

제조업체 부품 F9PH5 | Dell 부품 490-BEYM | 주문 코드 490-Beym | Dell

사용 GPU: T4 Tesla GPU



코드를 실행해 보자!

This notebook is open with private outputs. Outputs will not be saved. You can disable this in [Notebook settings](#).

ds-ai-test.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Table of contents

- Test for DS/AI class
- Section

### Test for DS/AI class

Server spec!

```
[1] 1 from tensorflow.python.client import device_lib
    2 device_lib.list_local_devices()

[{'name': '/device:CPU:0',
  device_type: 'CPU',
  memory_limit: 268435456,
  locality {
  },
  incarnation: 17670787319713240863, name: '/device:GPU:0',
  device_type: 'GPU',
  memory_limit: 14638920512,
  locality {
    bus_id: 1
    links {
    }
  },
  incarnation: 9972398538379896466,
  physical_device_desc: "device: 0, name: Tesla T4, pci bus id: 0000:00:04.0, compute capability: 7.5"}]
```

```
[2] 1 import platform
    2 platform.platform()

'Linux-4.19.112+-x86_64-with-Ubuntu-18.04-bionic'
```

```
1 # write hello world!
2 print('hello world!')

hello world!
```



내 드라이브 - Google 드라이브 x +

drive.google.com/drive/u/0/my-drive

Apps 고려대학교 지식기... 데이터과학원 자료... 공유 드라이브 - G... seojksc Tutorials of Object... Better-Boy/books-f... bsharvey.github.io/... 고려대학교 도서관 ICODEBROKER :: [...]

드라이브 드라이브에서 검색

새로 만들기

우선순위

내 드라이브

- 연구
- Colab Notebooks
- test
- 공유 드라이브

공유 문서함

최근 문서함

중요 문서함

휴지통

내 드라이브

빠른 액세스

- 2021 DS&AI Syllabus  
신은경[ 조교수 / 사회학과 ]님이 오늘 ...
- Data Science & AI  
신은경[ 조교수 / 사회학과 ]님이 지난...
- 이력서-서정권.pdf  
지난주에 공유함
- 데이터과학과 인공지능 (가안...  
지난달에 열어봄

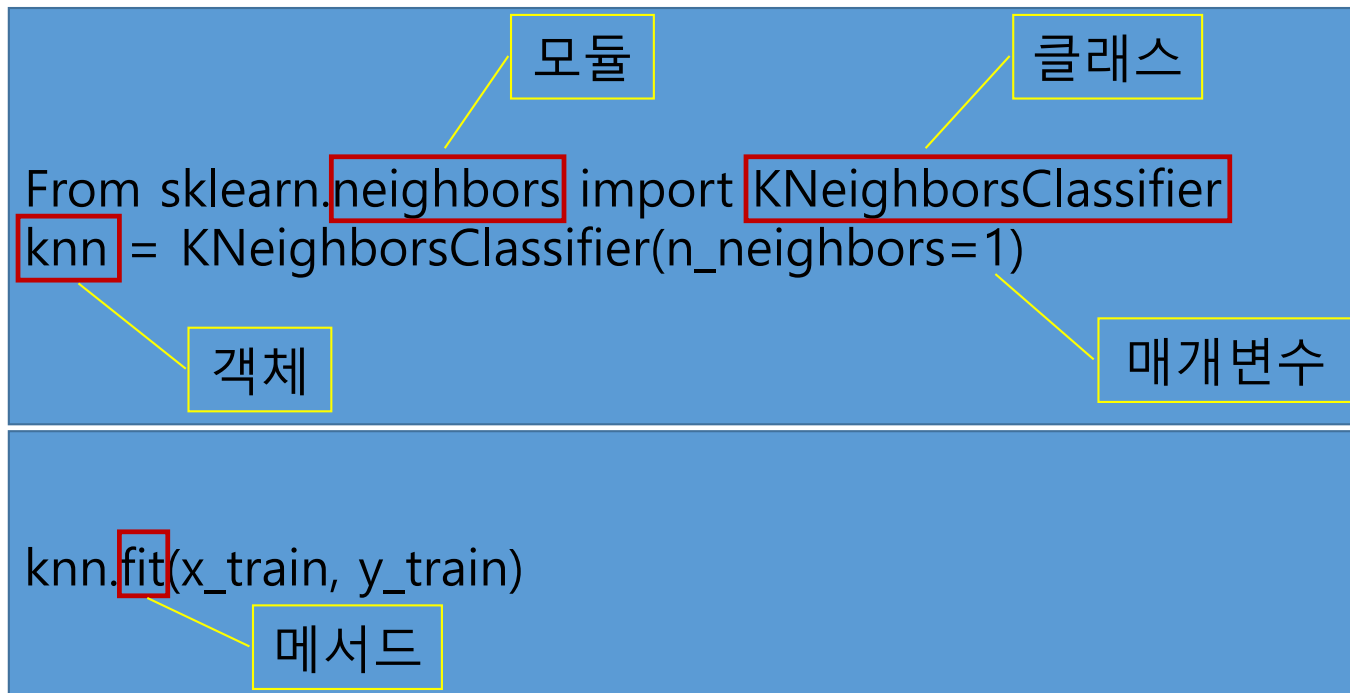
이름 ↑	소유자	마지막으로 수정한 날짜	파일 크기
연구	나	오후 12:44 나	—
Colab Notebooks	나	오후 1:04 나	—
test	나	오후 12:45 나	—
고려대학교 미래성장연구소_교육과정_중견기업 미래성장 ...	나	2020. 6. 12. 나	58MB

수업 실습 코드 다운  
로드 하기!  
(구글 드라이브 접속)



# Python 패키지, 모듈, 클래스

- 패키지(라이브러리): scikit-learn, NumPy, Scipy, matplotlib, pandas





# NumPy

- 파이썬 과학 계산 필수 패키지
- 다차원 배열과 선형 대수 연산과 푸리에 변환 등 수학 함수
- scikit-learn에서 NumPy 배열은 기본 데이터 구조
  - scikit-learn은 NumPy 배열 형태의 데이터를 입력으로 받음
- (<http://www.numpy.org/>) 참조

```
>>>import numpy as np
>>>x = np.array([[1,2,3], [4,5,6]])
>>>print("x=\n{}".format(x))
x=
[[1 2 3]
 [4 5 6]]
```



# SciPy

- 과학 계산용 함수를 모아놓은 파이썬 패키지
- 고성능 선형대수, 함수 최적화, 신호처리, 통계 분포 등 기능 제공
- (<https://www.scipy.org/scipylib>) 참조

```
>>>import numpy as np
>>>from scipy import sparse
>>>eye = np.eye(4)
>>>print(eye)
[[ 1.  0.  0.  0.]
 [ 0.  1.  0.  0.]
 [ 0.  0.  1.  0.]
 [ 0.  0.  0.  1.]]
```

```
>>>spmat = sparse.csr_matrix(eye)
>>>print(spmat)
(0, 0) 1.0
(1, 1) 1.0
(2, 2) 1.0
(3, 3) 1.0
```





# Scikit-learn

- 오픈소스(자유롭게 사용 및 배포 가능)
- 산업, 학계에서 널리 사용
- NumPy와 SciPy를 기반으로 개발됨
- 알고리즘을 설명한 풍부한 문서제공
  - <http://scikit-learn.org/stable/documentation>
- 설치 (무료 배포판)
  - Anaconda (<https://www.anaconda.com>) : 윈도우, 리눅스, 맥
  - Python(x,y) (<https://python-xy.github.io/>) : 윈도우



## Python 자료 구조 - array

**입력치** VECTOR:  $A = [1, 2, 3, 4, 5, 6, 7]$

**메모리** ADDRESS: 

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]
------	------	------	------	------	------	------

$A[:4] = [1, 2, 3, 4]$

A[0]	A[1]	A[2]	A[3]
------	------	------	------

$A[4:] = [5, 6, 7]$

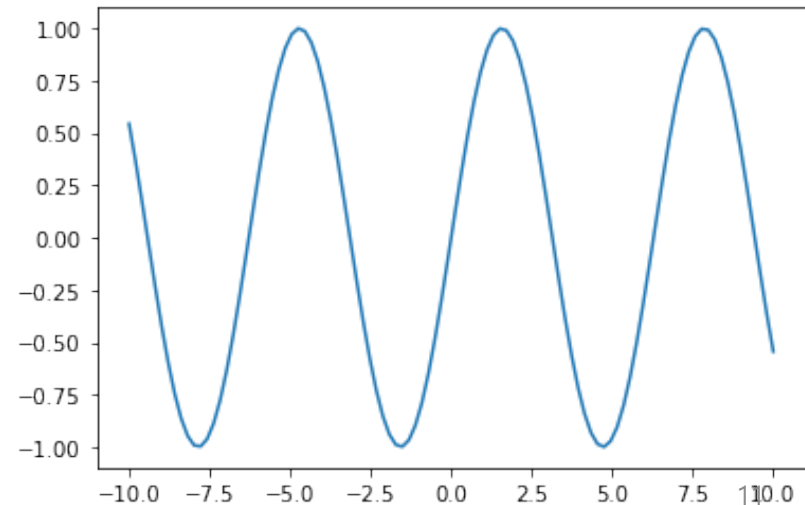
A[4]	A[5]	A[6]
------	------	------



# matplotlib

- 파이썬 과학 계산용 그래프 패키지
- 선 그래프, 히스토그램, 산점도 등을 지원
- 데이터와 분석 결과를 다양한 관점에서 시각화하는 것은 매우 중요
- 주피터 노트북에서 사용 시 %matplotlib inline 명령을 사용  
(또는 %matplotlib notebook)

```
%matplotlib inline
import matplotlib.pyplot as plt
x = np.linspace(-10, 10, 100)
y = np.sin(x)
plt.plot(x, y, marker="x")
```





# pandas

- 파이썬 데이터 처리와 분석을 위한 패키지
- R의 data.frame을 본떠서 설계한 DataFrame이라는 데이터 구조를 기반으로 개발됨
- pandas의 DataFrame은 엑셀의 스프레드시트와 비슷한 테이블 형태
- SQL, 엑셀 파일, CSV 파일 등 다양한 데이터를 읽을 수 있음

```
import pandas as pd
data = {"Name": ['Kim', 'Lee', 'Park'],
        "Location": ['Seoul', 'Busan', 'Deagu'],
        "Age": [24, 13, 50]}
data_pandas = pd.DataFrame(data)
display(data_pandas)
```

	Name	Location	Age
0	Kim	Seoul	24
1	Lee	Busan	13
2	Park	Deagu	50



# Scikit learn – MLP for MNIST

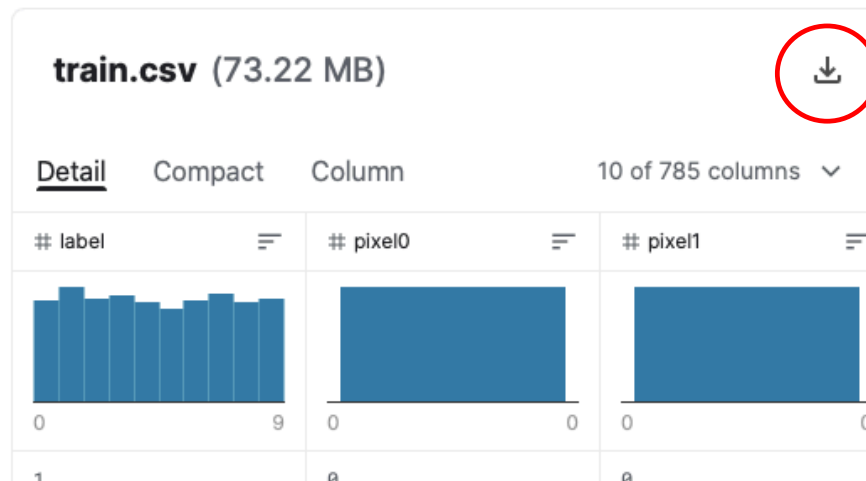
- Kaggle example
  - URL site: <https://www.kaggle.com/icinnamon/mnist-scikit-learn-tutorial?select=test.csv>
- Data download

## Input

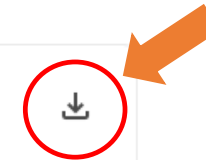
122.2 MB

### Data Sources

- Digit Recognizer
  - sample\_submission.csv
  - test.csv
  - train.csv



Click!





# Explore data by Pandas – head()

```
1 import csv as csv
2 import numpy as np
3 import pandas as pd
4 from sklearn.neural_network import MLPClassifier
5 from sklearn.metrics import accuracy_score
6 from sklearn.model_selection import train_test_split
7
8 train_df = pd.read_csv("../content/train.csv", header=0)
9 train_df.head()
```

pixel780	pixel781	pixel782	pixel783
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	pixel10	pixel11	pixel12	pixel13
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

5 rows x 785 columns



## Create Test/Train Data

```
import csv as csv
import numpy as np
import pandas as pd
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
```

```
train_df = pd.read_csv("../input/train.csv", header=0)
```

```
train_data = train_df.values
X_train, X_test, y_train, y_test = train_test_split(train_data[0::1:], train_data[0::0],
test_size=0.2, random_state=0)
```



## MLP Classifier

```
def __init__(hidden_layer_sizes=(100,), activation='relu', solver='adam',  
alpha=0.0001, batch_size='auto', learning_rate='constant',  
learning_rate_init=0.001, power_t=0.5, max_iter=200, shuffle=True,  
random_state=None, tol=0.0001, verbose=False, warm_start=False, momentum=0.9,  
nesterovs_momentum=True, early_stopping=False, validation_fraction=0.1,  
beta_1=0.9, beta_2=0.999, epsilon=1e-08, n_iter_no_change=10, max_fun=15000)
```

```
clf = MLPClassifier(solver='sgd')  
clf.fit(X_train, y_train)  
neural_output = clf.predict(X_test)  
print("sgd")  
print(accuracy_score(y_test, neural_output))
```

Default setting

```
clf = MLPClassifier(solver='lbfgs') ...
```





## Saving the Output

```
output = forest_output
predictions_file = open("forest_output.csv", "w")
open_file_object = csv.writer(predictions_file)
ids = range(forest_output.__len__())
ids = [x+1 for x in ids]
open_file_object.writerow(["ImageId", "Label"])
open_file_object.writerows(zip(ids, output))
predictions_file.close()
print('Saved "forest_output" to file.')
```

```
sgd 0.9226190476190477
Saved "neural_output" to file.
```

The screenshot shows a Jupyter Notebook titled 'ds-ai-test.ipynb'. The left sidebar displays a file explorer with a folder named 'sample\_data' containing two files: 'neural\_output.csv' (highlighted with an orange box) and 'train.csv'. The main area shows a code cell with the following Python code:

```
25 print("sgd")
26 print(accuracy_score(y_test, neural_output))
27
28 output = neural_output
29 predictions_file = open("neural_output.csv", "w")
30 open_file_object = csv.writer(predictions_file)
31 ids = range(neural_output.__len__())
32 ids = [x+1 for x in ids]
33 open_file_object.writerow(["ImageId", "Label"])
34 open_file_object.writerows(zip(ids, output))
35 predictions_file.close()
36 print('Saved "neural_output" to file.')
```

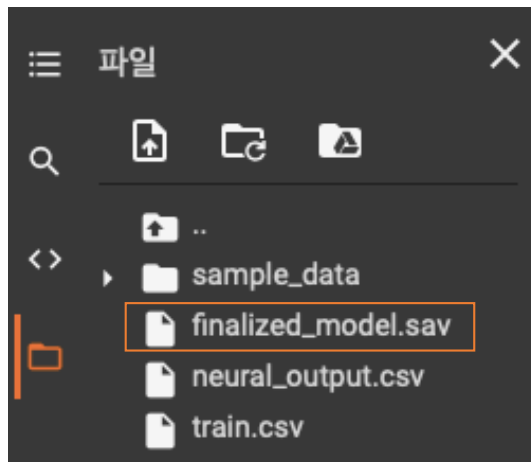


# Save the Model

```
import pickle

# save the model to disk
filename = 'finalized_model.sav'
pickle.dump(clf, open(filename, 'wb'))
```

Some time later!



```
# load the model from disk
filename = 'finalized_model.sav'
loaded_model = pickle.load(open(filename, 'rb'))
neural_output = loaded_model.predict(X_test)
print("sgd_from_saved_model")
print(accuracy_score(y_test, neural_output))
```



# Load the Model and Run

```
1 import csv as csv
2 import numpy as np
3 import pandas as pd
4 from sklearn.neural_network import MLPClassifier
5 from sklearn.metrics import accuracy_score
6 from sklearn.model_selection import train_test_split
7 import pickle
8
9 train_df = pd.read_csv("../content/train.csv", header=0)
10 train_data = train_df.values
11
12 X_train, X_test, y_train, y_test = train_test_split(train_data[0::,1::], train_data[0::,0], test_size=0.2, random_state=0)
13
14
15 # load the model from disk
16 filename = 'finalized_model.sav'
17 loaded_model = pickle.load(open(filename, 'rb'))
18 neural_output = loaded_model.predict(X_test)
19 print("sgd_from_saved_model")
20 print(accuracy_score(y_test, neural_output))
21
22
```

```
sgd_from_saved_model
0.9070238095238096
```



# References

- [https://colab.research.google.com/notebooks/intro.ipynb#scrollTo=5fCEDCU\\_qrC0](https://colab.research.google.com/notebooks/intro.ipynb#scrollTo=5fCEDCU_qrC0)
- <https://theorydb.github.io/dev/2019/08/23/dev-ml-colab/>
- <https://roboreport.co.kr/%EC%9B%B9%EC%97%90%EC%84%9C-%EA%B0%84%EB%8B%A8%ED%95%98%EA%B2%8C-%EB%94%A5%EB%9F%AC%EB%8B%9D-%EC%8B%A4%EC%8A%B5%ED%95%98%EA%B8%B0-%EA%B5%AC%EA%B8%80-colab-%EC%82%AC%EC%9A%A9%EB%B0%A9%EB%B2%95/>



**Thank you**



# APPENDIX.

## [Python modules for Machine Learning]

Normalization

`StandardScaler`

MNIST dataset

`from sklearn.datasets import fetch_openml`

Shuffling

`numpy.random.permutation`

Data split

`train_test_split`  
`test_size=0.3`



# Data Import

```
from sklearn.datasets import fetch_openml
mnist = fetch_openml('mnist_784')
mnist.data.shape, mnist.target.shape
# (70000, 784)
```



# Data Split 1

```
split_ratio = 0.9
n_train = int(mnist.data.shape[0] * split_ratio)
print(n_train)
# 63000
n_test = mnist.data.shape[0] - n_train
print(n_test)
# 7000
X_train = mnist.data[:n_train] y_train = mnist.target[:n_train]
print(X_train.shape, y_train.shape)
# ((63000, 784), (63000,))
X_test = mnist.data[n_train:] y_test = mnist.target[n_train:]
print(X_test.shape, y_test.shape)
# ((7000, 784), (7000,))
```





# Data Split 2

```
import numpy as np from sklearn.model_selection import train_test_split
```

```
# generate samples
```

```
sample = np.arange(100)
```

```
print(sample)
```

```
# array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, # 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, # 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, # 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, # 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, # 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99])
```

```
print(train_test_split(sample))
```

```
# [array([46, 66, 42, 54, 58, 90, 11, 93, 28, 97, 17, 31, 55, 27, 74, 25, 91, # 8, 57, 9, 86, 39, 53, 73, 98, 44, 60, 43, 12, 82, 69, 2, 89, 83, # 10, 61, 0, 59, 99, 16, 88, 71, 68, 36, 20, 80, 76, 41, 30, 18, 22, # 75, 34, 50, 79, 37, 78, 52, 32, 14, 63, 92, 87, 5, 21, 24, 38, 72, # 96, 35, 51, 33, 94, 4, 65]), # array([84, 26, 6, 1, 62, 81, 1 5, 19, 29, 23, 7, 56, 77, 45, 49, 95, 3, # 85, 67, 13, 70, 40, 48, 64, 47])]
```

```
X_train, X_test = train_test_split(sample)
```

```
print(X_train.shape, X_test.shape)
```

```
# ((75,), (25,))
```



# RandomForest

```
from sklearn.ensemble import RandomForestClassifier
# module loading
clf = RandomForestClassifier()
# train data!
clf.fit(X_train, y_train)
# make prediction
prediction = clf.predict(X_test)
print(prediction.shape)
# 7000 # accuracy
result = (prediction == y_test).mean()
print(result)
# 0.9617142857142857
```



# Visualization

```
import matplotlib.pyplot as plt
```

```
%matplotlib inline
```

```
# 랜덤하게 몇 가지 data 가져오기
```

```
random_pick = np.random.randint(low=0, high=n_test, size=10)
```

```
random_pick
```

```
# array([3898, 6815, 6640, 2924, 451, 2688, 633, 6563, 5993, 4024])
```

```
figure = plt.figure()
```

```
figure.set_size_inches(12, 5)
```

```
axes = []
```

```
for i in range(1, 11):
```

```
axes.append(figure.add_subplot(2, 5, i))
```

```
tmp_list = []
```

```
for i in range(10):
```

```
tmp = mnist.data[n_train + random_pick[i]]
```

```
tmp = tmp.reshape(-1, 28)
```

```
tmp_list.append(tmp)
```

```
print(y_test[random_pick])
```

```
for i in range(10):
```

```
axes[i].matshow(tmp_list[i])
```

```
# ['8' '6' '7' '2' '7' '7' '8' '8' '4' '3']
```

