

MNIST using PCA and kNN

PinkLAB Edu

11 November, 2025

Table of Contents

1	MNIST	3
1.1	NIST	3
1.2	MNIST	3
1.3	kaggle에서 받기.....	4
2	using PCA and kNN.....	5
2.1	데이터 읽기	5
2.2	train 데이터의 생긴 모양	5
2.3	test 데이터의 생긴 모양.....	6
2.4	데이터 정리	6
2.5	근데 어떻게 생긴 데이터인지 확인해볼까.....	7
2.6	random하게 16개만~.....	8
2.7	일단 fit	9
2.8	test 데이터 predict	9
2.9	kNN은 차원의 저주가 있다~~~	10
2.10	PCA로 차원을 줄여주자.....	10
2.11	best score	10
2.12	단지 이정도 수준으로 약 93%의 acc가 확보된다	11
2.13	결과확인.....	11
2.14	골고루 잘 맞추고 있다.....	11
2.15	숫자를 다시 확인하고 싶다면	12
2.16	틀린 데이터가 어떻게 생겼는지 확인해보자	12
2.17	틀린 데이터를 추려서	13
2.18	틀릴만 한 것도 있다	14

1 MNIST

1.1 NIST

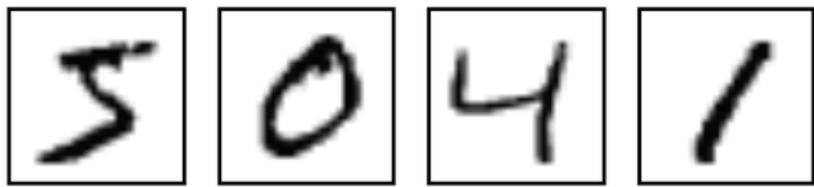
NAME	DATE	CITY	STATE ZIP	
[REDACTED]	8-3-89	Minden City	MI 48458	
This sample of handwriting is being collected for use in testing computer recognition of hand printed numbers and letters. Please print the following characters in the boxes that appear below.				
0 1 2 3 4 5 6 7 8 9	0 1 2 3 4 5 6 7 8 9	0 1 2 3 4 5 6 7 8 9		
0123456789	0123456789	0123456789		
87	701	3752	80759	960941
158	4586	32123	832656	82
7481	80539	419219	67	904
7481	80539	419219	67	904

- NIST 데이터 셋 (National Institute of Standards and Technology)

1.2 MNIST

0 0 0 0 0 0 0 0 0 0 0 0 0
 1 1 1 1 1 1 1 1 1 1 1 1 1
 2 2 2 2 2 2 2 2 2 2 2 2 2
 3 3 3 3 3 3 3 3 3 3 3 3 3
 4 4 4 4 4 4 4 4 4 4 4 4 4
 5 5 5 5 5 5 5 5 5 5 5 5 5
 6 6 6 6 6 6 6 6 6 6 6 6 6
 7 7 7 7 7 7 7 7 7 7 7 7 7
 8 8 8 8 8 8 8 8 8 8 8 8 8
 9 9 9 9 9 9 9 9 9 9 9 9 9

- MNIST 데이터 셋 (Modified National Institute of Standards and Technology)



- 28*28 픽셀의 0~9 사이의 숫자 이미지와 레이블로 구성된 셋
- 머신 러닝을 공부하는 사람들이 입문용으로 사용함
- 60000개의 훈련용 셋과 10000개의 실험용 셋으로 구성

1.3 kaggle에서 받기

- <https://www.kaggle.com/oddrationale/mnist-in-csv>

A screenshot of a Kaggle dataset page for "MNIST in CSV". The page title is "Dataset MNIST in CSV". Below the title is a preview of the CSV file content showing various digits. A user profile picture and name "Dariel Dato-on" are shown, with a note that it was updated 2 years ago (Version 2). Below the preview are navigation tabs: Data (highlighted), Tasks, Kernels (156), Discussion (1), Activity, and Metadata. To the right of these tabs are links for "Download (122 MB)" and "New Notebook". A red arrow points to the "Download" button. Below the tabs, there are sections for "Usability 8.2", "License CC0: Public Domain", and "Tags computer science, image data, beginner". The main content area contains a bold heading "The MNIST dataset provided in a easy-to-use CSV format" and a paragraph explaining that the original dataset is difficult for beginners and provides a CSV version by Joseph Redmon. It also lists the two files: "mnist_train.csv" and "mnist_test.csv".

2 using PCA and kNN

2.1 데이터 읽기

```
import pandas as pd

df_train = pd.read_csv('./data/mnist_train.csv')
df_test = pd.read_csv('./data/mnist_test.csv')

df_train.shape, df_test.shape
```

✓ 2.0s

Python

((60000, 785), (10000, 785))

2.2 train 데이터의 생긴 모양

df_train																
	label	1x1	1x2	1x3	1x4	1x5	1x6	1x7	1x8	1x9	...	28x19	28x20	28x21	28x22	28x23
0	5	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	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
4	9	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
...
59995	8	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
59996	3	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
59997	5	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
59998	6	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
59999	8	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0

60000 rows × 785 columns

2.3 test 데이터의 생긴 모양

df_test															Python	
	label	1x1	1x2	1x3	1x4	1x5	1x6	1x7	1x8	1x9	...	28x19	28x20	28x21	28x22	28x23
0	7	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
1	2	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	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
4	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
...
9995	2	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
9996	3	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
9997	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
9998	5	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
9999	6	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0

10000 rows × 785 columns

2.4 데이터 정리

```
import numpy as np

X_train = np.array(df_train.iloc[:, 1:])
y_train = np.array(df_train['label'])

X_test = np.array(df_test.iloc[:, 1:])
y_test = np.array(df_test['label'])

X_train.shape , y_train.shape, X_test.shape, y_test.shape
```

Python

((60000, 784), (60000,), (10000, 784), (10000,))

2.5 근데 어떻게 생긴 데이터인지 확인해볼까

```
import random

samples = random.choices(population = range(0, 60000), k = 16)
samples
```

✓ 0.0s

```
[1564,  
 54334,  
 15404,  
 27256,  
 18357,  
 8537,  
 25855,  
 34053,  
 11067,  
 21952,  
 3399,  
 31435,  
 8632,  
 18167,  
 15316,  
 1220]
```

Python

2.6 random하게 16개만~

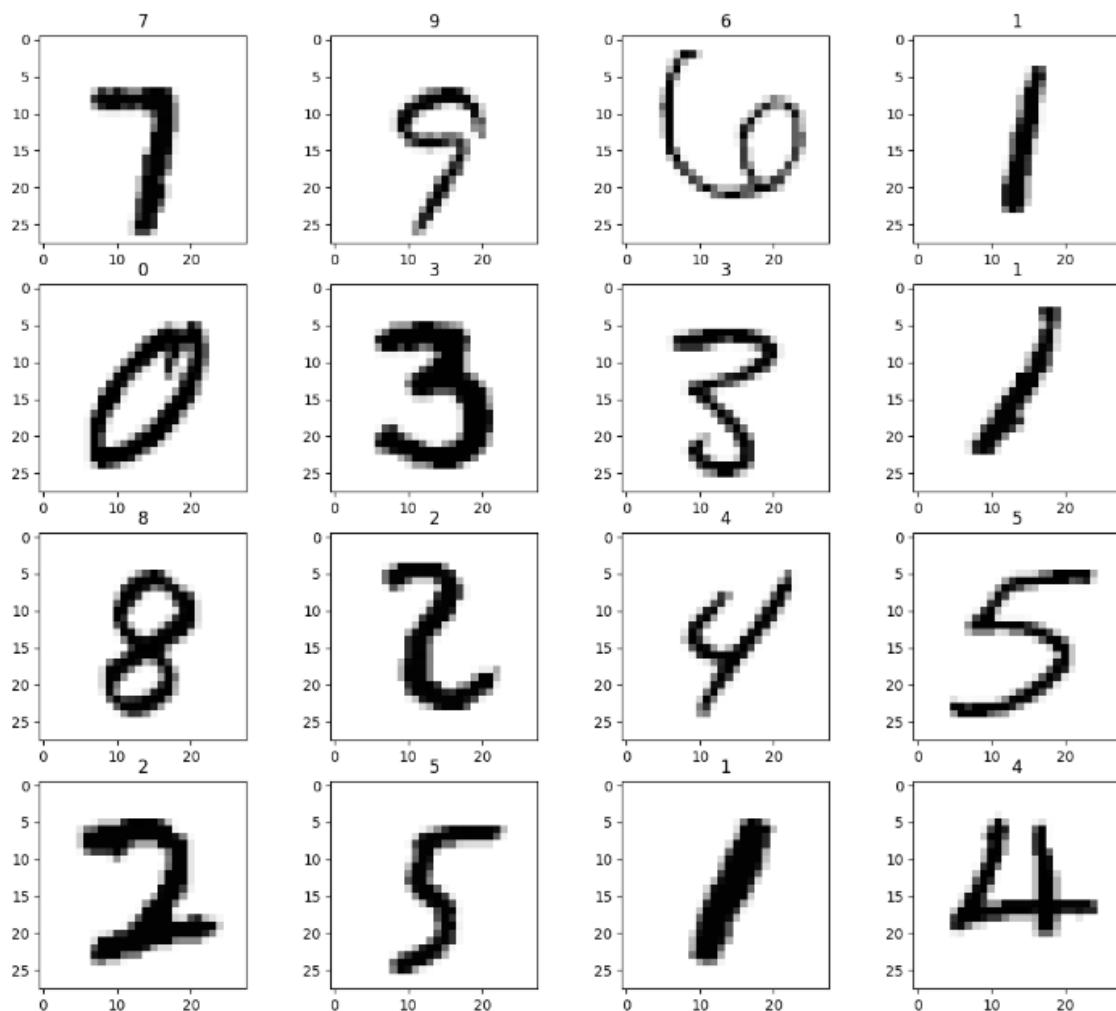
```
import matplotlib.pyplot as plt

plt.figure(figsize=(14, 12))

for idx, n in enumerate(samples):
    plt.subplot(4, 4, idx + 1)
    plt.imshow(X_train[n].reshape(28, 28), cmap="Greys", interpolation="nearest")
    plt.title(y_train[n])

plt.show()
```

✓ 1.1s Python



2.7 일단 fit

```
from sklearn.neighbors import KNeighborsClassifier  
import time  
  
start_time = time.time()  
  
clf = KNeighborsClassifier(n_neighbors=5)  
clf.fit(X_train, y_train)  
  
print("Fit time : ", time.time() - start_time)
```

✓ 0.7s

Python

Fit time : 0.13118720054626465

2.8 test 데일리 predict

```
from sklearn.metrics import accuracy_score  
  
start_time = time.time()  
pred = clf.predict(X_test)  
print("Fit time : ", time.time() - start_time)  
print(accuracy_score(y_test, pred))
```

✓ 13.3s

Python

Fit time : 13.377159118652344

0.9688

2.9 kNN은 차원의 저주가 있다~~~

2.10 PCA로 차원을 줄여주자

```
from sklearn.pipeline import Pipeline
from sklearn.decomposition import PCA
from sklearn.model_selection import GridSearchCV, StratifiedKFold

pipe = Pipeline(
    [
        ("pca", PCA()),
        ("clf", KNeighborsClassifier()),
    ]
)

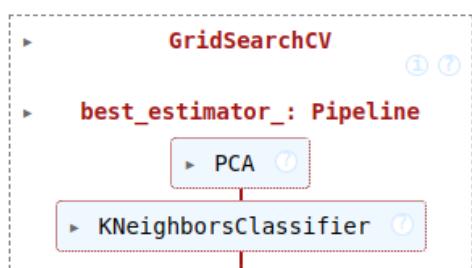
parameters = {"pca_n_components": [2, 5, 10], "clf_n_neighbors": [5, 10, 15]}

kf = StratifiedKFold(n_splits=5, shuffle=True, random_state=13)
grid = GridSearchCV(pipe, parameters, cv=kf, n_jobs=-1, verbose=1)
grid.fit(X_train, y_train)
```

✓ 1m 4.2s

Python

Fitting 5 folds for each of 9 candidates, totalling 45 fits



2.11 best score

```
print("Best score : %.3f" % grid.best_score_)
print("Best parameters set : ")

best_parameters = grid.best_estimator_.get_params()
for param_name in sorted(parameters.keys()):
    print("\t%s: %r" % (param_name, best_parameters[param_name]))
```

✓ 0.0s

Python

```
Best score : 0.931
Best parameters set :
    clf_n_neighbors: 10
    pca_n_components: 10
```

2.12 단지 이정도 수준으로 약 93%의 acc가 확보된다

```
accuracy_score(y_test, grid.best_estimator_.predict(X_test))
```

Python

0.9291

2.13 결과확인

```
def results(y_pred, y_test):
    from sklearn.metrics import classification_report

    print(classification_report(y_test, y_pred))

results(grid.predict(X_train), y_train)
```

Python

2.14 골고루 잘 맞추고 있다

	precision	recall	f1-score	support
0	0.96	0.98	0.97	5923
1	0.98	0.99	0.98	6742
2	0.96	0.96	0.96	5958
3	0.94	0.90	0.92	6131
4	0.94	0.93	0.93	5842
5	0.93	0.94	0.93	5421
6	0.96	0.98	0.97	5918
7	0.96	0.95	0.96	6265
8	0.92	0.91	0.91	5851
9	0.90	0.91	0.90	5949
accuracy			0.94	60000
macro avg	0.94	0.94	0.94	60000
weighted avg	0.94	0.94	0.94	60000

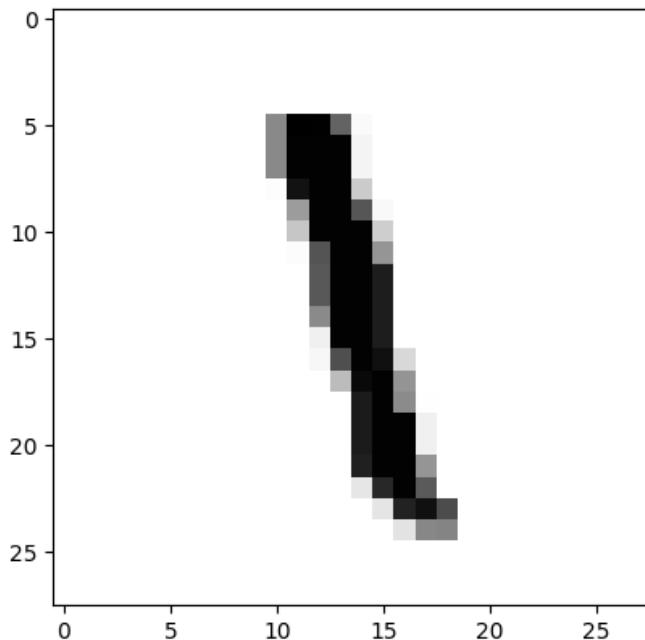
2.15 숫자를 다시 확인하고 싶다면

```
n = 700
plt.imshow(X_test[n].reshape(28, 28), cmap="Greys", interpolation="nearest")
plt.show()

print("Answer is : ", grid.best_estimator_.predict(X_test[n].reshape(1, 784)))
print("Real Label is : ", y_test[n])
```

✓ 0.0s

Python



```
Answer is : [1]
Real Label is : 1
```

2.16 틀린 데이터가 어떻게 생겼는지 확인해보자

```
preds = grid.best_estimator_.predict(X_test)
preds
```

✓ 1.2s

Python

```
array([7, 2, 1, ..., 4, 5, 6], shape=(10000,))
```

```
y_test
```

✓ 0.0s

Python

```
array([7, 2, 1, ..., 4, 5, 6], shape=(10000,))
```

2.17 틀린 데이터를 추려서

```
wrong_results = X_test[y_test != preds]
samples = random.choices(population=range(0, wrong_results.shape[0]), k=16)

plt.figure(figsize=(14, 12))

for idx, n in enumerate(samples):
    plt.subplot(4, 4, idx + 1)
    plt.imshow(
        wrong_results[n].reshape(28, 28),
        cmap="Greys",
        interpolation="nearest"
    )
    plt.title(grid.best_estimator_.predict(wrong_results[n].reshape(1, 784))[0])

plt.show()
```

Python

2.18 틀릴만 한 것도 있다

