zero-base /

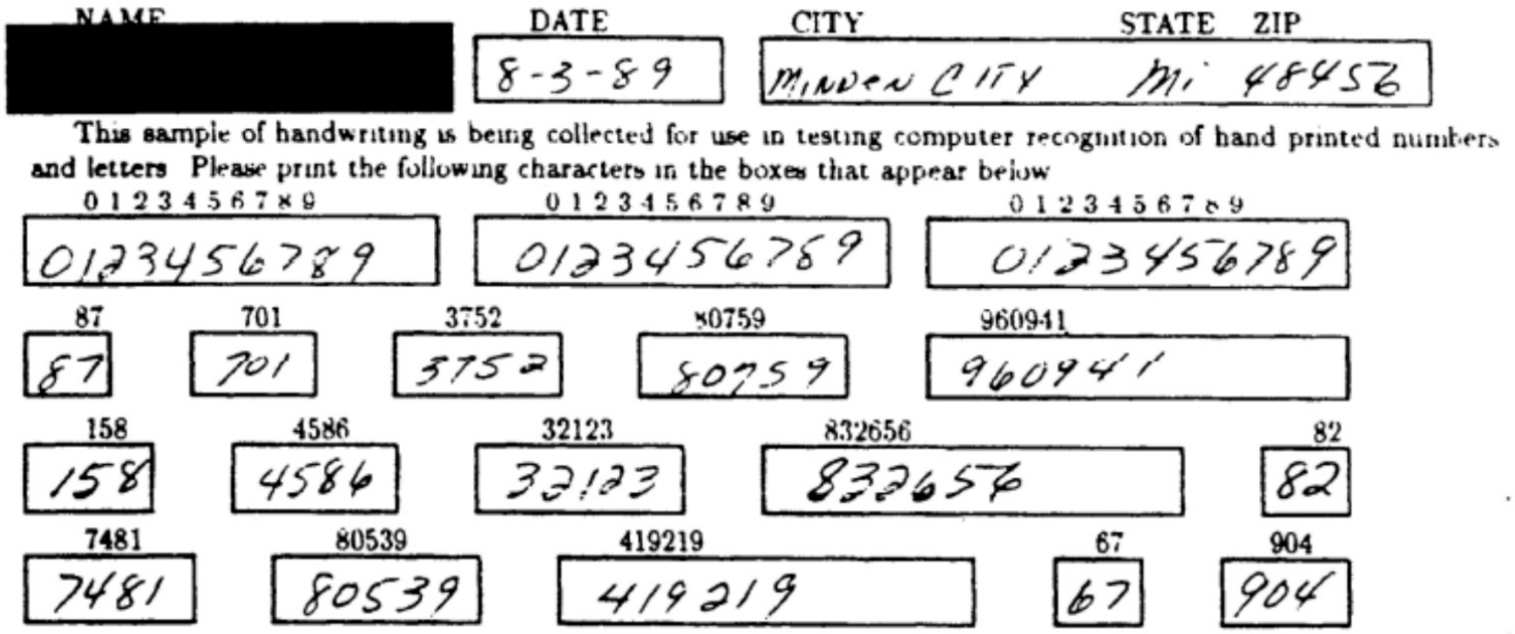
Life Changing Education

Chapter 25. MNIST data using PCA and kNN

MNIST

zero-base /

NIST


NAME [redacted] DATE 8-3-89 CITY MINDEN CITY STATE MI ZIP 48456

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 0123456789 0123456789 0123456789

87 701 3752 80759 960941

158 4586 32123 832656 82

7481 80539 419219 67 904

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

zero-base /

MNIST

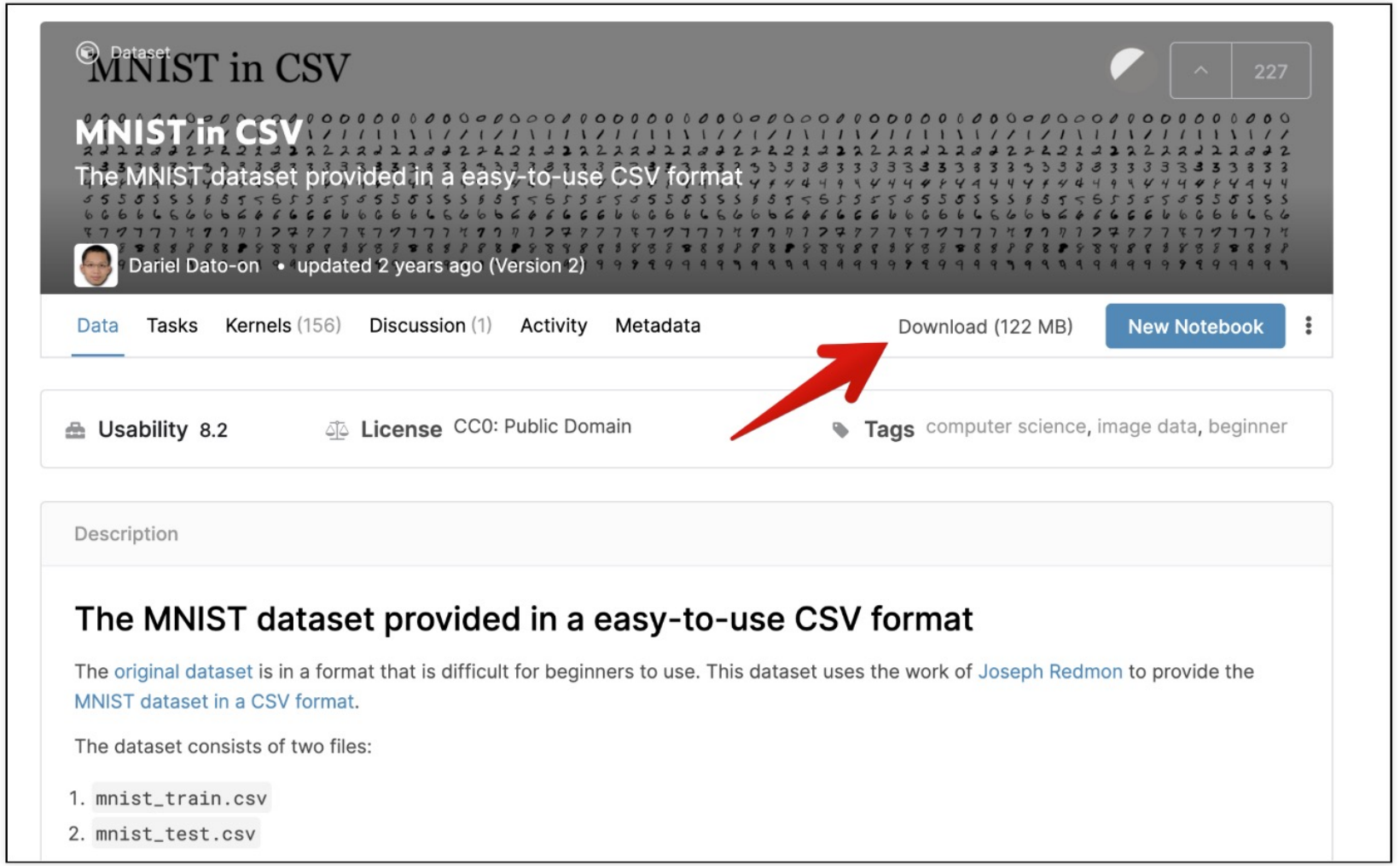
- **MNIST** 데이터 셋 **(Modified National Institute of Standards and Technology)**

zero-base /

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

zero-base /

kaggle에서 받기



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

using PCA and kNN

zero-base /

데이터 읽기

```
import pandas as pd

df_train = pd.read_csv('./mnist-in-csv/mnist_train.csv')
df_test = pd.read_csv('./mnist-in-csv/mnist_test.csv')

df_train.shape, df_test.shape
```

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

zero-base /

train 데이터의 생긴 모양

```
df_train.head()
```

	label	1x1	1x2	1x3	1x4	1x5	1x6	1x7	1x8	1x9	...	28x19	28x20	28x21	28x22	28x23	28x24	28x25
0	5	0	0	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	0	0
2	4	0	0	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	0	0
4	9	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0

5 rows × 785 columns

zero-base /

test 데이터의 생긴 모양

```
df_test
```

	label	1x1	1x2	1x3	1x4	1x5	1x6	1x7	1x8	1x9	...	28x19	28x20	28x21	28x22	28x23	28x24	28x2
0	7	0	0	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	0	0
2	1	0	0	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	0	0
4	4	0	0	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	0	0
9996	3	0	0	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	0	0
9998	5	0	0	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	0	0

```
10000 rows × 785 columns
```

zero-base /

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
```

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

zero-base /

근데 어떻게 생긴 데이터인지 확인해볼까

```
import random

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

```
[52171,
 47114,
 8583,
 13948,
 9334,
 35152,
 59638,
 20999,
 30436,
```

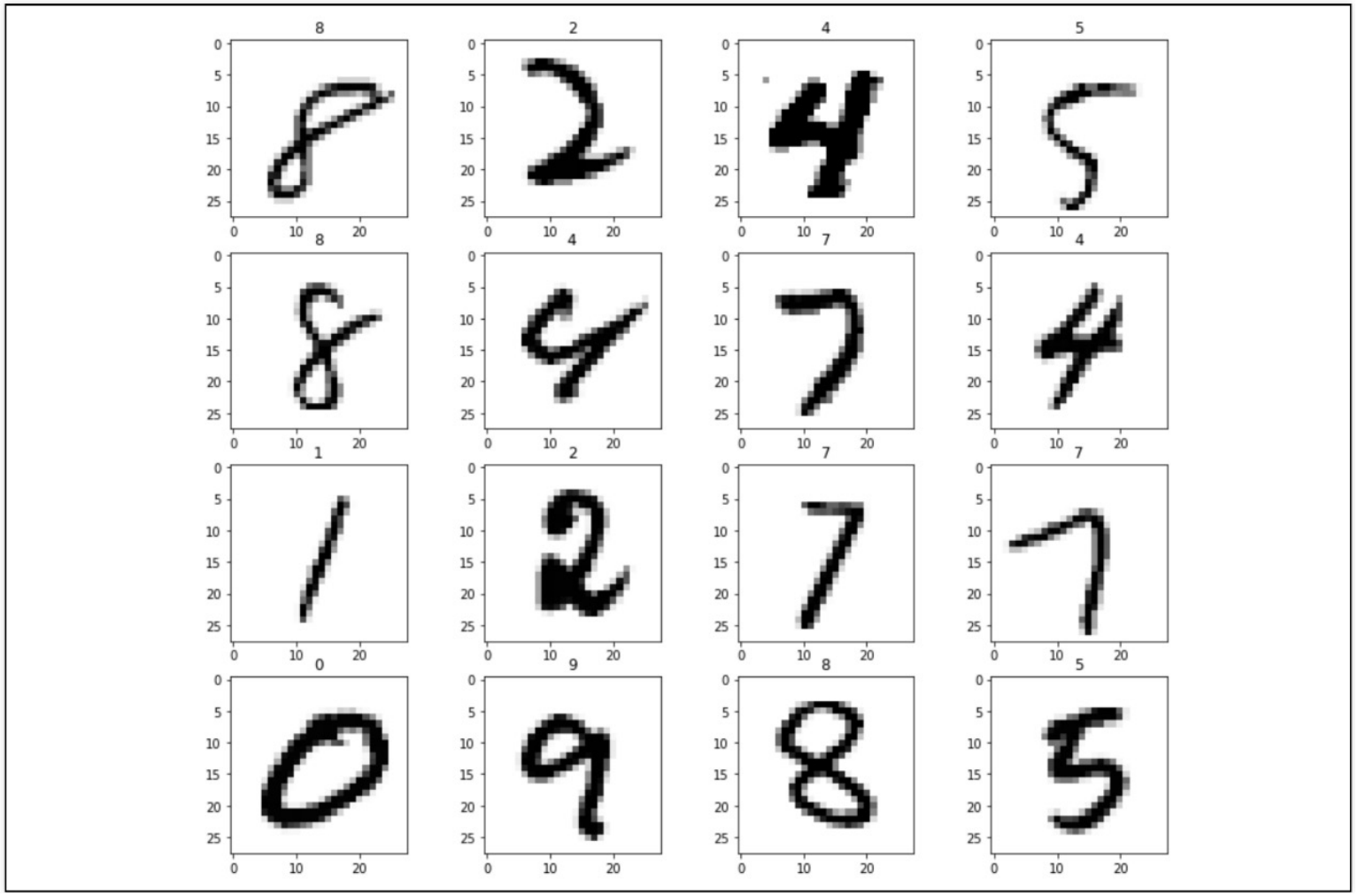
zero-base /

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()
```

zero-base /

이렇게 생겼다



zero-base /

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)
```

```
Fit time :  13.366422891616821
```

zero-base /

test 데이터 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))
```

```
Fit time :  702.2971158027649
0.9688
```

- 저 시간 실화??

kNN은 차원의 저주가 있다~~~

zero-base /

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)
```

kNN은 차원의 저주가 있다~~~

best score

```
print("Best score: %0.3f" % 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]))
```

```
Best score: 0.931
Best parameters set:
        clf__n_neighbors: 10
        pca__n_components: 10
```

zero-base /

단지 이정도 수준으로 약 93%의 acc가 확보된다

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

```
0.9286
```

결과확인

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

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

zero-base /

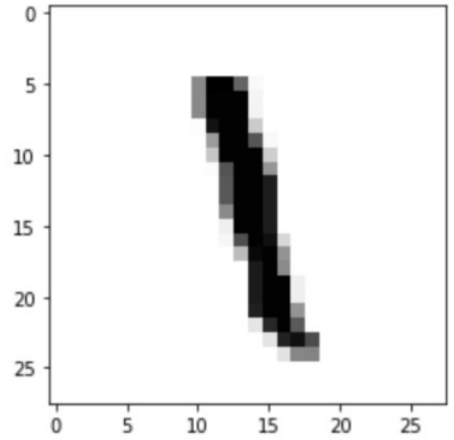
골고루 잘 맞추고 있다

	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			0.95	60000
macro avg	0.94	0.94	0.94	60000
weighted avg	0.94	0.95	0.94	60000

zero-base /

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])
```




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

kNN은 차원의 저주가 있다~~~

틀린 데이터가 어떻게 생겼는지 확인해보자

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

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

```
y_test
```

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

zero-base /

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()
```

kNN은 차원의 저주가 있다~~~

틀릴만 한 것도 있다