In [3]:
```python
#데이터 처리
import pandas as pd
import numpy as np

#데이터 시각화
import matplotlib.pyplot as plt
import seaborn as sns

#이미지 데이터셋
from sklearn import datasets

#데이터 정제
from sklearn.preprocessing import StandardScaler

#정확도 (분류)
from sklearn.metrics import accuracy_score

#2차원
from sklearn.manifold import TSNE

#군집합
from sklearn.cluster import KMeans

#분류 기계 학습
from sklearn.ensemble import RandomForestClassifier
```

와인 품질 세분화하기

In [4]:
```python
#데이터 불러오기
df = pd.read_csv("/Users/youngjinseo/Desktop/python/wine-clustering.csv")
df.head(8)
```

Out[4]:

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonflavanoid_Phenols	Proanthocyanins	Color_Intensity	Hue	OD280	Proline
0	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065
1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050
2	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185
3	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480
4	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735
5	14.20	1.76	2.45	15.2	112	3.27	3.39	0.34	1.97	6.75	1.05	2.85	1450
6	14.39	1.87	2.45	14.6	96	2.50	2.52	0.30	1.98	5.25	1.02	3.58	1290
7	14.06	2.15	2.61	17.6	121	2.60	2.51	0.31	1.25	5.05	1.06	3.58	1295

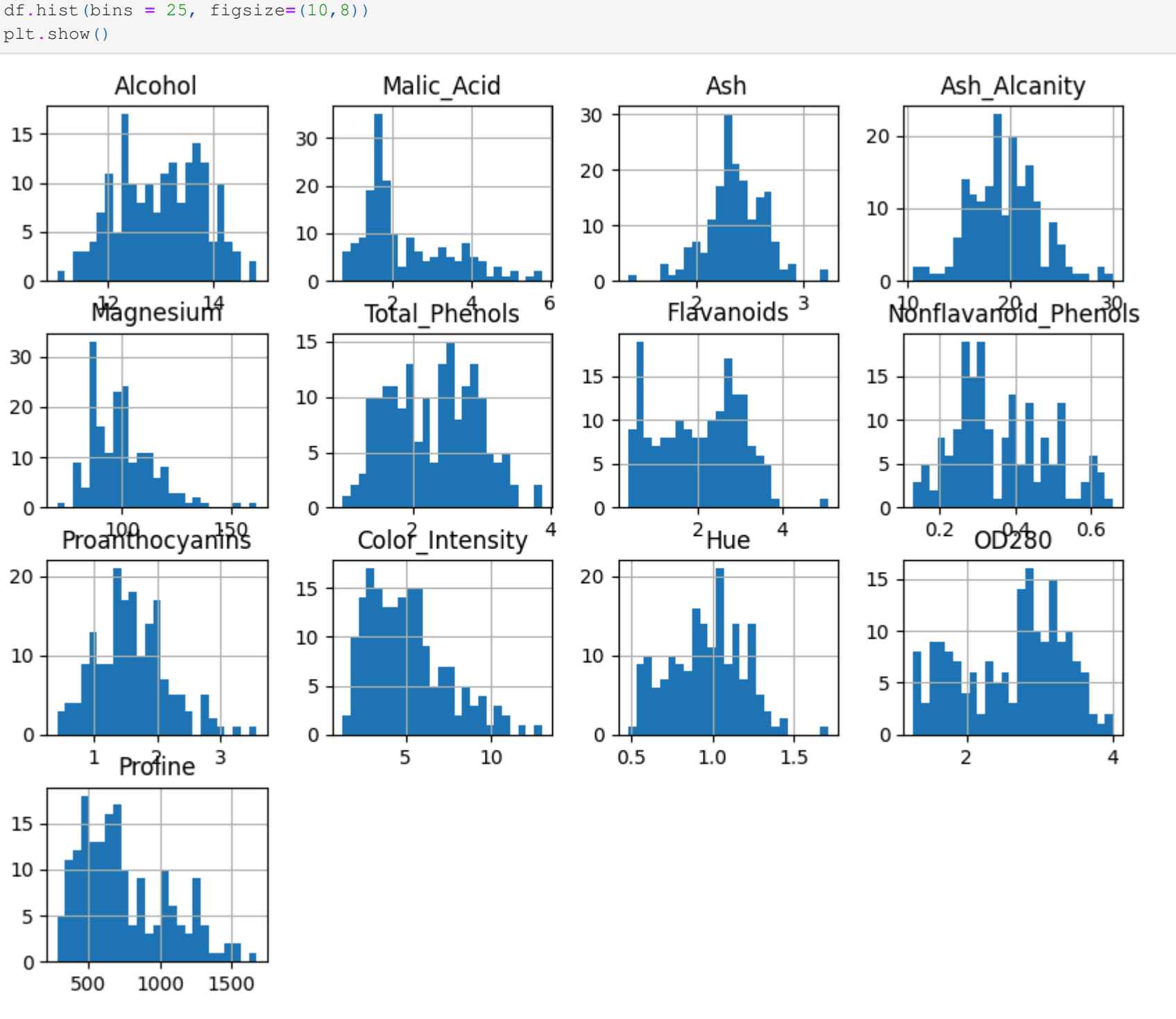
In [5]:
```python
#데이터 정보
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Alcohol               178 non-null    float64
 1   Malic_Acid            178 non-null    float64
 2   Ash                   178 non-null    float64
 3   Ash_Alcanity          178 non-null    float64
 4   Magnesium             178 non-null    int64  
 5   Total_Phenols         178 non-null    float64
 6   Flavanoids            178 non-null    float64
 7   Nonflavanoid_Phenols  178 non-null    float64
 8   Proanthocyanins       178 non-null    float64
 9   Color_Intensity       178 non-null    float64
 10  Hue                   178 non-null    float64
 11  OD280                 178 non-null    float64
 12  Proline               178 non-null    int64  
dtypes: float64(11), int64(2)
memory usage: 18.2 KB
```

In [6]:
```python
#데이터 결측값(데이터 없는 거) 개수
df.isnull().sum()
```

Out[6]:
```
Alcohol                 0
Malic_Acid              0
Ash                     0
Ash_Alcanity            0
Magnesium               0
Total_Phenols           0
Flavanoids              0
Nonflavanoid_Phenols    0
Proanthocyanins         0
Color_Intensity         0
Hue                     0
OD280                   0
Proline                 0
dtype: int64
```

In [7]:
```python
#히스토그램으로 질 분포 보기
df.hist(bins = 25, figsize=(10,8))
plt.show()
```



In [9]:
```python
# StandardScaler 객체를 생성함. StandardScaler는 데이터를 표준화하여 평균이 0, 표준편차가 1이 되도록 변환함.
scaler = StandardScaler()

# fit_transform은 데이터를 변환하기 위해 각 특성의 평균과 표준편차를 학습(fit)하고, 이를 사용해 변환(transform)함.
scaler_train = scaler.fit_transform(df)


# 표준화된 데이터를 다시 데이터프레임 형태로 변환하여 wine이라는 새로운 데이터프레임을 만듦.
# columns=df.columns를 통해 원본 데이터프레임과 같은 열 이름을 유지함.
wine = pd.DataFrame(scaler_train, columns = df.columns)
wine.head(8)
```

Out[9]:

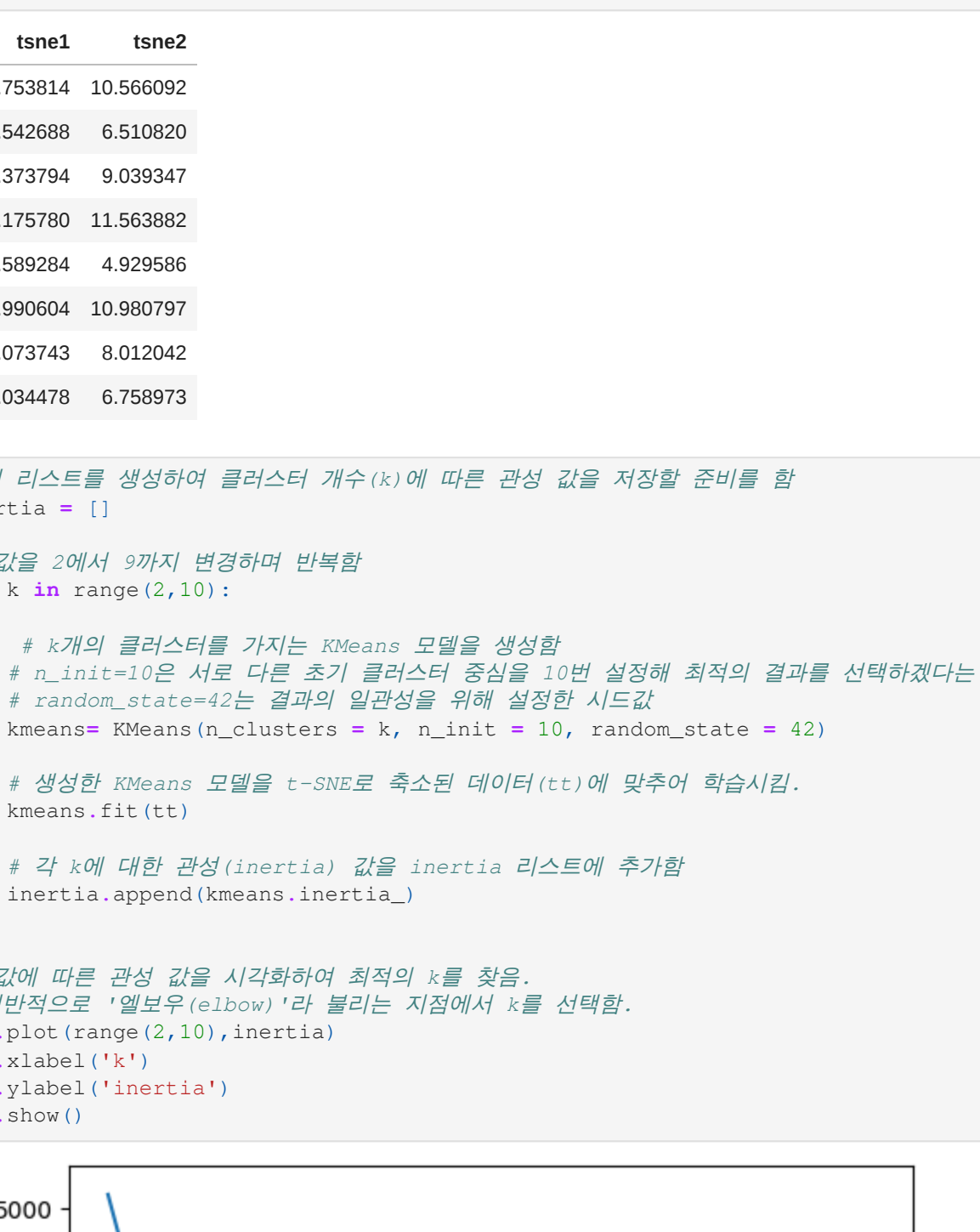
	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonflavanoid_Phenols	Proanthocyanins	Color_Intensity	Hue	OD280	Proline
0	1.518613	-0.562250	0.232053	-1.169593	1.913905	0.808997	1.034819	-0.659563	1.224884	0.251717	0.362177	1.847920	1.013009
1	0.246290	-0.499413	-0.827996	-2.490847	0.018145	0.568648	0.733629	-0.820719	-0.544721	-0.293321	0.406051	1.113449	0.965242
2	0.196879	0.021231	1.109334	-0.268738	0.088358	0.808997	1.215533	-0.498407	2.135968	0.269020	0.318304	0.788587	1.395148
3	1.691550	-0.346811	0.487926	-0.809251	0.930918	2.491446	1.466525	-0.981875	1.032155	1.186068	-0.427544	1.184071	2.334574
4	0.295700	0.227694	1.840403	0.451946	1.281985	0.808997	0.663351	0.226796	0.401404	-0.319276	0.362177	0.449601	-0.037874
5	1.481555	-0.517367	0.305159	-1.289707	0.860705	1.562093	1.366128	-0.176095	0.664217	0.731870	0.406051	0.336606	2.239039
6	1.716255	-0.418624	0.305159	-1.469878	-0.262708	0.328298	0.492677	-0.498407	0.681738	0.083015	0.274431	1.367689	1.729520
7	1.308617	-0.167278	0.890014	-0.569023	1.492625	0.488531	0.482637	-0.417829	-0.597284	-0.003499	0.449924	1.367689	1.745442

In [10]:
```python
# TSNE 객체를 생성함 . n_components=2는 데이터를 2차원으로 축소하겠다는 의미이고,
# random_state=42는 결과의 재현성을 위해 랜덤 시드를 설정함
tsne = TSNE(n_components = 2, random_state = 42)
tsne_train = tsne.fit_transform(wine)

#t-SNE로 축소된 2차원 데이터를 새로운 데이터프레임(tt)으로 만듦.
# 열 이름은 'tsne1', 'tsne2'로 설정하여 시각화에 활용하기 쉽게 함.

tt = pd.DataFrame(tsne_train, columns = ['tsne1','tsne2'])
tt.head(8)
```

Out[10]:

	tsne1	tsne2
0	1.753814	10.566092
1	2.542688	6.510820
2	6.373794	9.039347
3	4.175780	11.563882
4	7.589284	4.929586
5	4.990604	10.980797
6	4.073743	8.012042
7	7.034478	6.758973

In [11]:
```python
# 빈 리스트를 생성하여 클러스터 개수(k)에 따른 관성 값을 저장할 준비를 함
inertia = []

# k값을 2에서 9까지 변경하며 반복함
for k in range(2,10):

     # k개의 클러스터를 가지는 KMeans 모델을 생성함
    # n_init=10은 서로 다른 초기 클러스터 중심을 10번 설정해 최적의 결과를 선택하겠다는 의미이고,
    # random_state=42는 결과의 일관성을 위해 설정한 시드값
    kmeans= KMeans(n_clusters = k, n_init = 10, random_state = 42)

    # 생성한 KMeans 모델을 t-SNE로 축소된 데이터(tt)에 맞추어 학습시킴.
    kmeans.fit(tt)

    # 각 k에 대한 관성(inertia) 값을 inertia 리스트에 추가함
    inertia.append(kmeans.inertia_)


# k값에 따른 관성 값을 시각화하여 최적의 k를 찾음.
# 일반적으로 '엘보우(elbow)'라 불리는 지점에서 k를 선택함.
plt.plot(range(2,10),inertia)
plt.xlabel('k')
plt.ylabel('inertia')
plt.show()
```

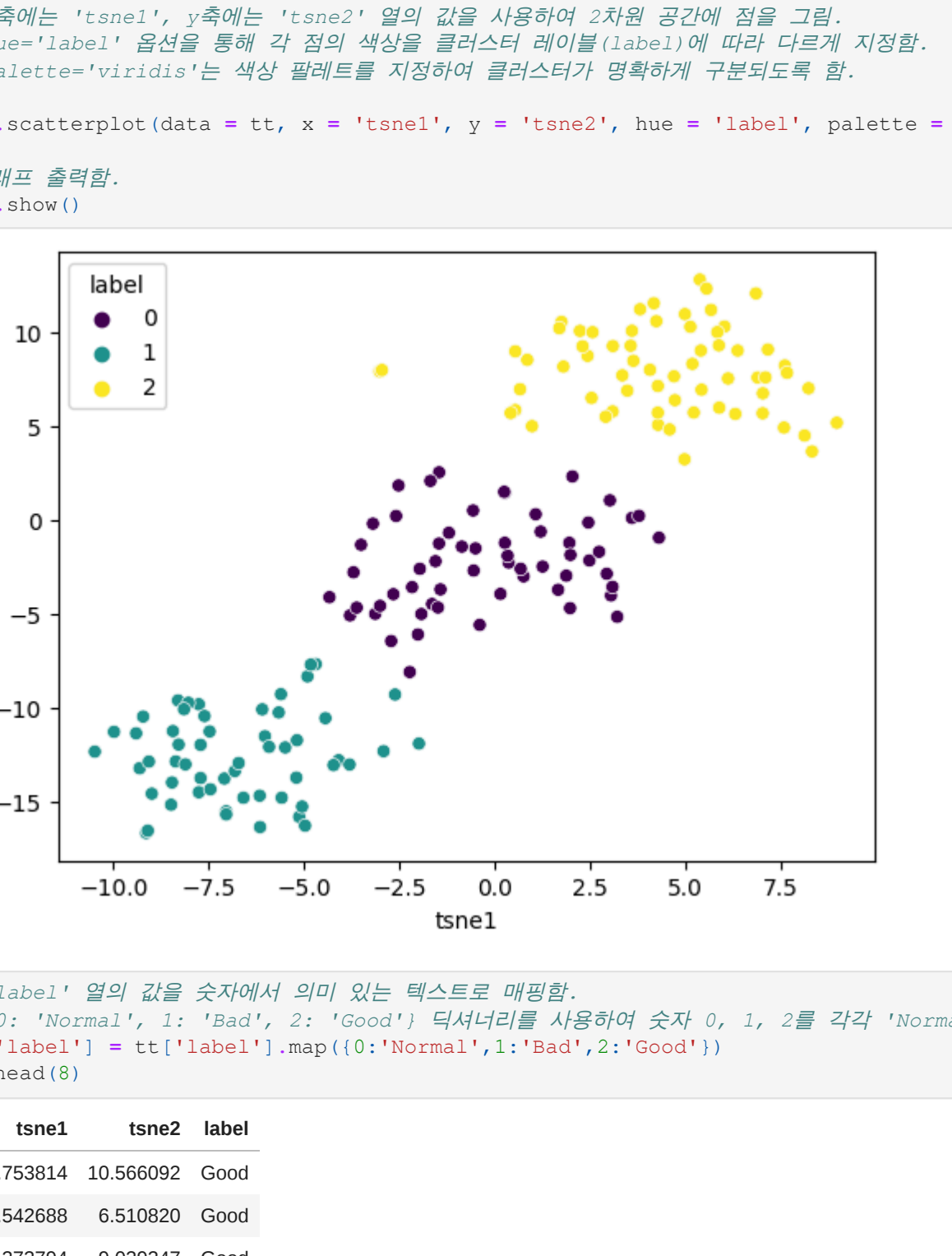


In [12]:
```python
# 클러스터 개수(n_clusters)를 3으로 설정하여 KMeans 모델을 생성함.
# n_init=10은 초기 클러스터 중심 설정을 10번 시도하여 최적의 결과를 선택하겠다는 의미.
# random_state=42를 통해 동일한 결과를 얻기 위한 시드를 설정함.
kmeans= KMeans(n_clusters = 3, n_init = 10, random_state = 42)

# fit_predict 메서드를 사용해 tt 데이터프레임에 클러스터링을 수행하고,
# 각 데이터 포인트에 대해 클러스터 레이블을 예측하여 'label' 열에 저장함
tt['label'] = kmeans.fit_predict(tt)


tt.head(8)
```

Out[12]:

	tsne1	tsne2	label
0	1.753814	10.566092	2
1	2.542688	6.510820	2
2	6.373794	9.039347	2
3	4.175780	11.563882	2
4	7.589284	4.929586	2
5	4.990604	10.980797	2
6	4.073743	8.012042	2
7	7.034478	6.758973	2

In [14]:
```python
# tt 데이터프레임을 사용하여 산점도를 생성함.
# x축에는 'tsne1', y축에는 'tsne2' 열의 값을 사용하여 2차원 공간에 점을 그림.
# hue='label' 옵션을 통해 각 점의 색상을 클러스터 레이블(label)에 따라 다르게 지정함.
# palette='viridis'는 색상 팔레트를 지정하여 클러스터가 명확하게 구분되도록 함.

sns.scatterplot(data = tt, x = 'tsne1', y = 'tsne2', hue = 'label', palette = 'viridis')

#그래프 출력함.
plt.show()
```

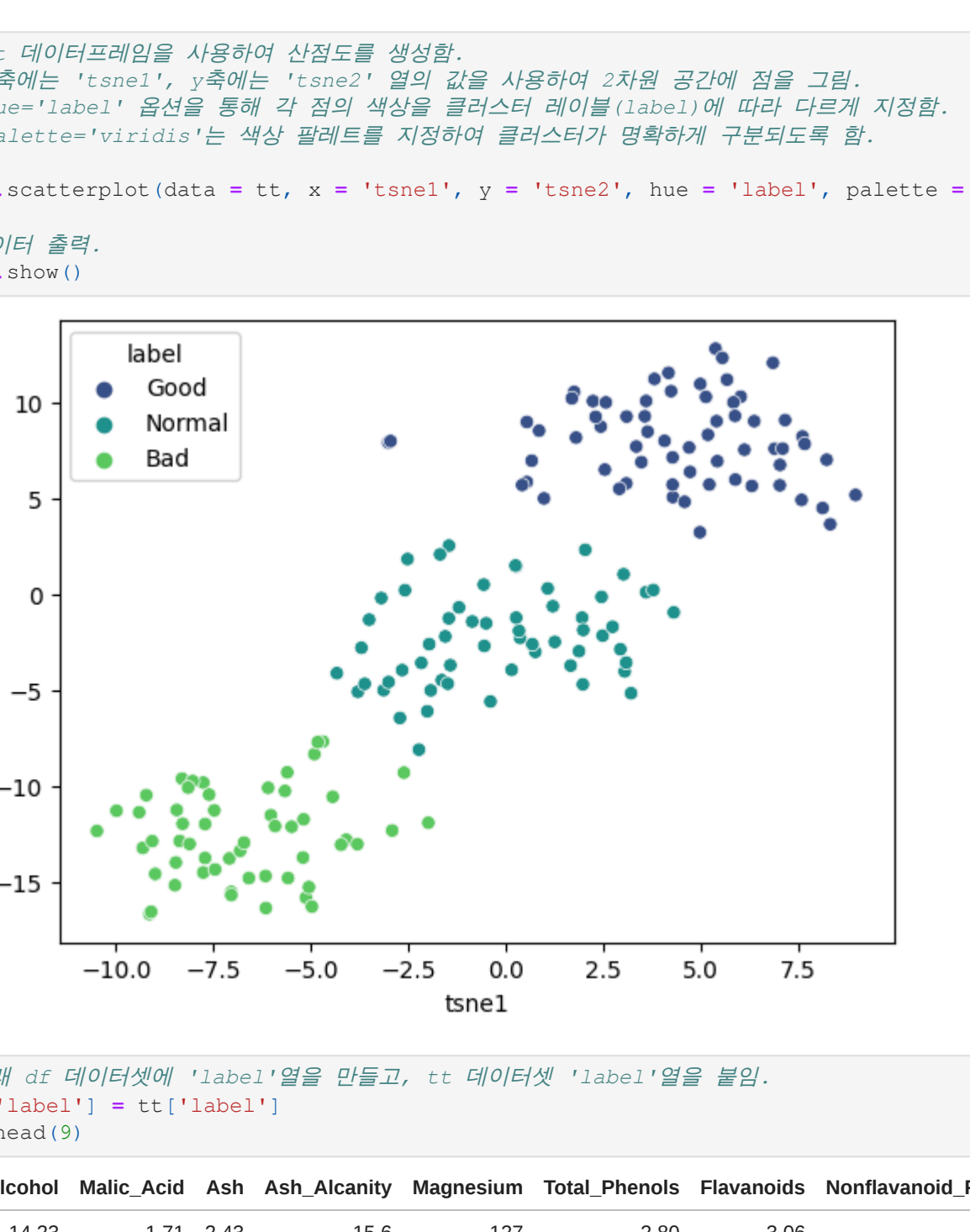


In [15]:
```python
# 'label' 열의 값을 숫자에서 의미 있는 텍스트로 매핑함.
# {0: 'Normal', 1: 'Bad', 2: 'Good'} 딕셔너리를 사용하여 숫자 0, 1, 2를 각각 'Normal', 'Bad', 'Good'으로 변환.
tt['label'] = tt['label'].map({0:'Normal',1:'Bad',2:'Good'})
tt.head(8)
```

Out[15]:

	tsne1	tsne2	label
0	1.753814	10.566092	Good
1	2.542688	6.510820	Good
2	6.373794	9.039347	Good
3	4.175780	11.563882	Good
4	7.589284	4.929586	Good
5	4.990604	10.980797	Good
6	4.073743	8.012042	Good
7	7.034478	6.758973	Good

In [16]:
```python
# tt 데이터프레임을 사용하여 산점도를 생성함.
# x축에는 'tsne1', y축에는 'tsne2' 열의 값을 사용하여 2차원 공간에 점을 그림.
# hue='label' 옵션을 통해 각 점의 색상을 클러스터 레이블(label)에 따라 다르게 지정함.
# palette='viridis'는 색상 팔레트를 지정하여 클러스터가 명확하게 구분되도록 함.

sns.scatterplot(data = tt, x = 'tsne1', y = 'tsne2', hue = 'label', palette = 'viridis')

#데이터 출력.
plt.show()
```



In [18]:
```python
#원래 df 데이터셋에 'label'열을 만들고, tt 데이터셋 'label'열을 붙임.
df['label'] = tt['label']
df.head(9)
```

Out[18]:

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonflavanoid_Phenols	Proanthocyanins	Color_Intensity	Hue	OD280	Proline	label
0	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065	Good
1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050	Good
2	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185	Good
3	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480	Good
4	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735	Good
5	14.20	1.76	2.45	15.2	112	3.27	3.39	0.34	1.97	6.75	1.05	2.85	1450	Good
6	14.39	1.87	2.45	14.6	96	2.50	2.52	0.30	1.98	5.25	1.02	3.58	1290	Good
7	14.06	2.15	2.61	17.6	121	2.60	2.51	0.31	1.25	5.05	1.06	3.58	1295	Good
8	14.83	1.64	2.17	14.0	97	2.80	2.98	0.29	1.98	5.20	1.08	2.85	1045	Good

In [19]:
```python
#Bad 관련 데이터 검색
df[df['label'] == 'Bad'].head(8)
```

Out[19]:

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonflavanoid_Phenols	Proanthocyanins	Color_Intensity	Hue	OD280	Proline	label
60	12.33	1.10	2.28	16.0	101	2.05	1.09	0.63	0.41	3.27	1.250	1.67	680	Bad
61	12.64	1.36	2.02	16.8	100	2.02	1.41	0.53	0.62	5.75	0.980	1.59	450	Bad
68	13.34	0.94	2.36	17.0	110	2.53	1.30	0.55	0.42	3.17	1.020	1.93	750	Bad
70	12.29	1.61	2.21	20.4	103	1.10	1.02	0.37	1.46	3.05	0.906	1.82	870	Bad
83	13.05	3.86	2.32	22.5	85	1.65	1.59	0.61	1.62	4.80	0.840	2.01	515	Bad
96	11.81	2.12	2.74	21.5	134	1.60	0.99	0.14	1.56	2.50	0.950	2.26	625	Bad
118	12.77	3.43	1.98	16.0	80	1.63	1.25	0.43	0.83	3.40	0.700	2.12	372	Bad
130	12.86	1.35	2.32	18.0	122	1.51	1.25	0.21	0.94	4.10	0.760	1.29	630	Bad

손글씨 분류

In [21]:
```python
#이미지 데이터셋 불러오기.
digits = datasets.load_digits()
```

In [22]:
```python
# 데이터셋의 키를 확인.
digits.keys()
```

Out[22]:
```
dict_keys(['data', 'target', 'frame', 'feature_names', 'target_names', 'images', 'DESCR'])
```

In [24]:
```python
# 네 번째 손글씨 숫자 이미지를 시각화.
plt.imshow(digits.images[3])
plt.show()
```

In [26]:
```python
# 4x5 형태의 서브플롯을 생성하여 총 20개의 이미지를 출력할 준비함.
fig, axes = plt.subplots(4,5,figsize = (10,8))


# 첫 20개의 이미지를 반복하여 출력함.
for i in range(20):
    ae = axes[i//5,i%5]# 현재 위치에 해당하는 서브플롯을 선택함
    ae.imshow(digits.images[i])#각 손글씨 이미지를 출력함
    ae.axis('off')# 축을 표시하지 않도록 설정함

#이미지 출력.
plt.show()
```

In [28]:
```python
#모델 함수 만들기.
model = RandomForestClassifier()
```

In [29]:
```python
#x는 digits의 data, y는 digits의 target으로 나누기.
x = np.array(digits.data)
y = np.array(digits.target)
```

In [30]:
```python
from sklearn.model_selection import train_test_split

# x는 특징(feature) 데이터, y는 레이블(target) 데이터
# train_test_split을 사용하여 데이터를 학습 세트와 테스트 세트로 나눔.
xtrain,xtest,ytrain,ytest = train_test_split(x,y, test_size = 0.2 , random_state = 42)
```

In [31]:
```python
#xtrain하고 ytrain학습 시키기
model.fit(xtrain,ytrain)
```

Out[31]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

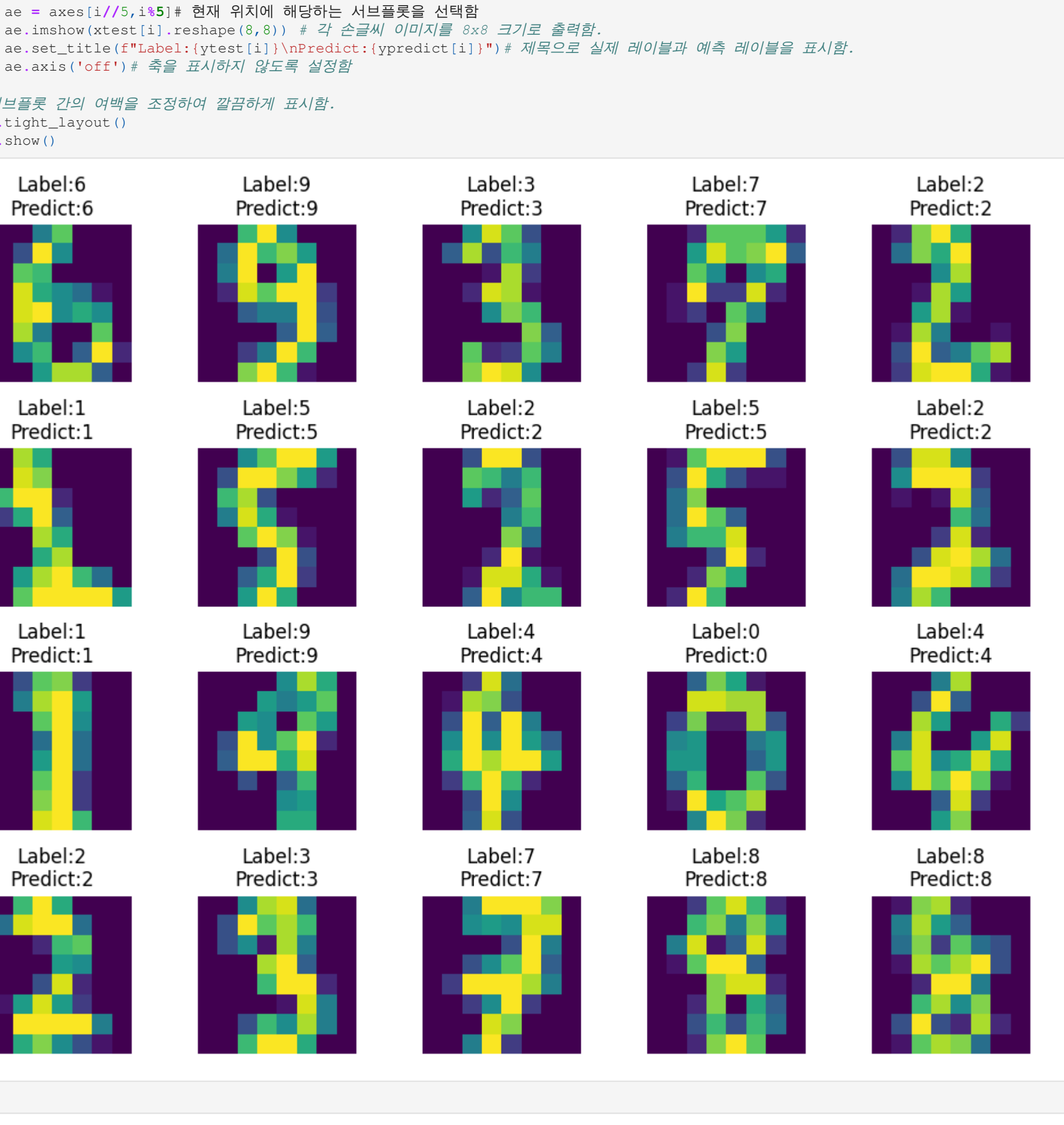
In [32]:
```python
# xtest: 테스트 세트의 입력 데이터
# ytest: 테스트 세트의 실제 레이블 데이터
#평가하기
model.score(xtest,ytest)
```

Out[32]:
```
0.9805555555555555
```

In [33]:
```python
#xtest 예측하기
ypredict = model.predict(xtest)

#ytest와 ypredict 정확도 알기
accuracy = accuracy_score(ytest,ypredict)
print(f"정확도는?:{accuracy*100:2f}%")
```

```
정확도는?:98.055556%
```

In [36]:
```python
# 4x5 형태의 서브플롯을 생성하여 총 20개의 이미지를 출력할 준비함.
fig, axes = plt.subplots(4,5,figsize = (10,8))

# 첫 20개의 이미지를 반복하여 출력함.
for i in range(20):
    ae = axes[i//5,i%5]# 현재 위치에 해당하는 서브플롯을 선택함
    ae.imshow(xtest[i].reshape(8,8)) # 각 손글씨 이미지를 8x8 크기로 출력함.
    ae.set_title(f"Label:{ytest[i]}\nPredict:{ypredict[i]}")# 제목으로 실제 레이블과 예측 레이블을 표시함.
    ae.axis('off')# 축을 표시하지 않도록 설정함

# 서브플롯 간의 여백을 조정하여 깔끔하게 표시함.
plt.tight_layout()
plt.show()
```



In [ ]: