```
# IMPORTANT: SOME KAGGLE DATA SOURCES ARE PRIVATE
# RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES.
import kagglehub
kagglehub.login()
```

```
# IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES,
# THEN FEEL FREE TO DELETE THIS CELL.
# NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS FROM KAGGLE'S PYTHON
# ENVIRONMENT SO THERE MAY BE MISSING LIBRARIES USED BY YOUR
# NOTEBOOK.

digit_recognizer_path = kagglehub.competition_download('digit-recognizer')

print('Data source import complete.')
```

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/digit-recognizer/sample_submission.csv
/kaggle/input/digit-recognizer/train.csv
/kaggle/input/digit-recognizer/test.csv
```

```
df=pd.read_csv('/kaggle/input/digit-recognizer/train.csv')
```

```
df.sample(5)
```

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781
18496	8	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
10790	9	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
25266	5	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
18843	6	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
11269	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0

5 rows × 785 columns

```
x=df.iloc[:,1:]
y=df.iloc[:,0]
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

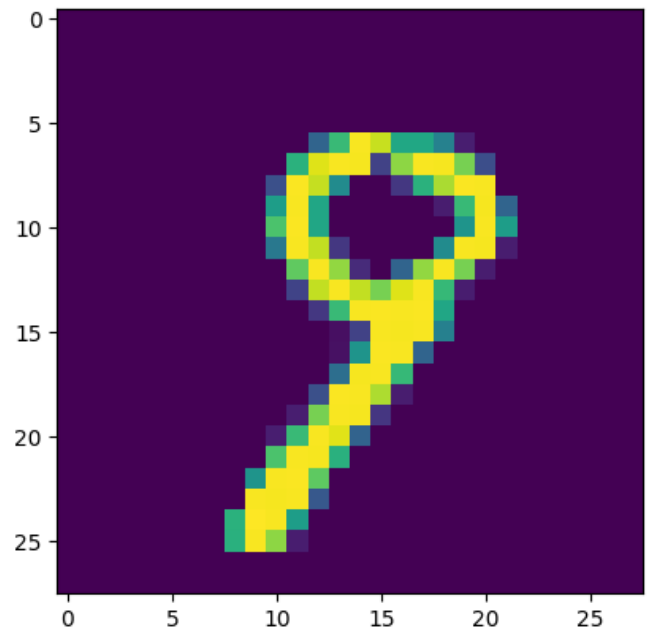
```
x_train.sample(5)
```

	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel774	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781
33701	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
614	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
6812	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
21890	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
17909	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0

5 rows × 784 columns

```
import matplotlib.pyplot as plt
plt.imshow(df.iloc[614,1:].values.reshape(28,28))
```

```
<matplotlib.image.AxesImage at 0x7821238903d0>
```




```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.transform(x_test)
```

```
from sklearn.decomposition import PCA
pca=PCA(n_components=None)
x_train=pca.fit_transform(x_train)
x_test=pca.transform(x_test)
```

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
knn.fit(x_train,y_train)
```

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

```
y_pred=knn.predict(x_test)
```

```
from sklearn.metrics import accuracy_score
accuracy_score()
```

```
array([3, 6, 9, ..., 2, 7, 2])
```

```
Start coding or generate with AI.
```