

```
In [29]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

import warnings
warnings.filterwarnings('ignore')

In [2]: train = pd.read_csv("D:\download\archive (4)\train.csv")
test = pd.read_csv("D:\download\archive (4)\test.csv")

In [3]: train.head()

Out[3]:
```

|   | label | pixel0 | pixel1 | pixel2 | pixel3 | pixel4 | pixel5 | pixel6 | pixel7 | pixel8 | ... | pixel774 | pixel775 | pixel776 | pixel777 | pixel778 | pixel779 | pixel780 | pixel781 | pixel782 | pixel783 |
|---|-------|--------|--------|--------|--------|--------|--------|--------|--------|--------|-----|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| 0 | 1     | 0      | 0      | 0      | 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        | 0        | 0        | 0        |
| 2 | 1     | 0      | 0      | 0      | 0      | 0      | 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        | 0        | 0        | 0        | 0        | 0        |
| 4 | 0     | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | ... | 0        | 0        | 0        | 0        | 0        | 0        | 0        | 0        | 0        | 0        |

5 rows × 785 columns

```
In [4]: test.head()

Out[4]:
```

|   | pixel0 | pixel1 | pixel2 | pixel3 | pixel4 | pixel5 | pixel6 | pixel7 | pixel8 | pixel9 | ... | pixel774 | pixel775 | pixel776 | pixel777 | pixel778 | pixel779 | pixel780 | pixel781 | pixel782 | pixel783 |
|---|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|-----|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| 0 | 0      | 0      | 0      | 0      | 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        | 0        | 0        | 0        |
| 2 | 0      | 0      | 0      | 0      | 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        | 0        | 0        | 0        |
| 4 | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | ... | 0        | 0        | 0        | 0        | 0        | 0        | 0        | 0        | 0        | 0        |

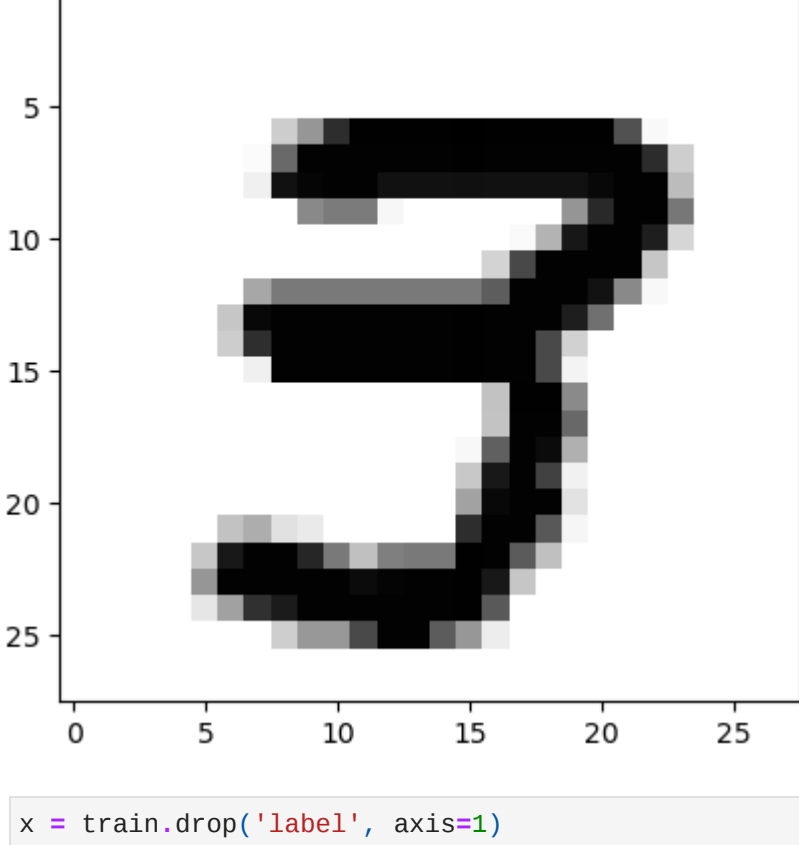
5 rows × 784 columns

```
In [7]: train.shape
test.shape

Out[7]: (28000, 784)

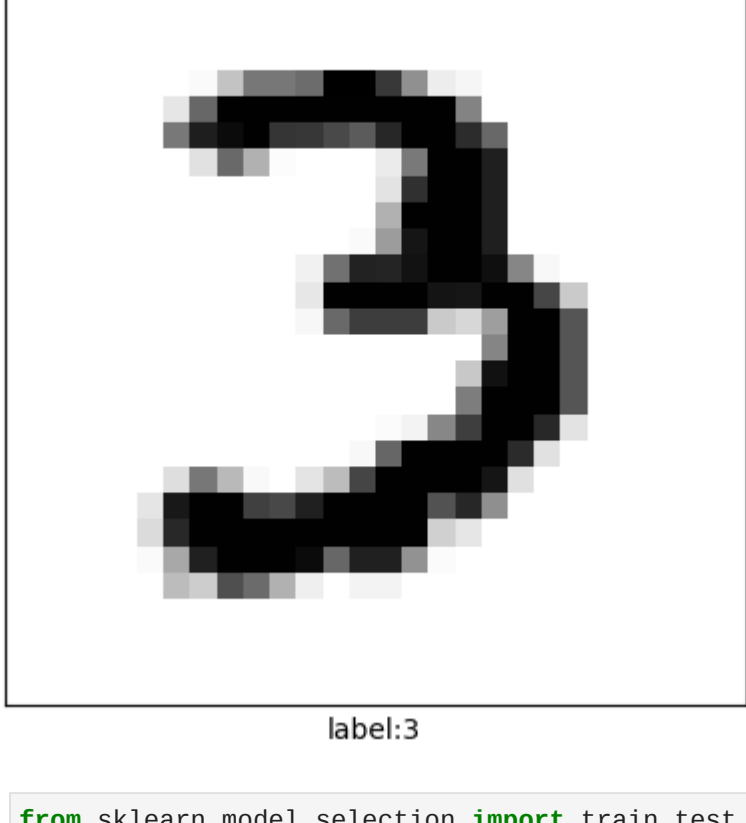
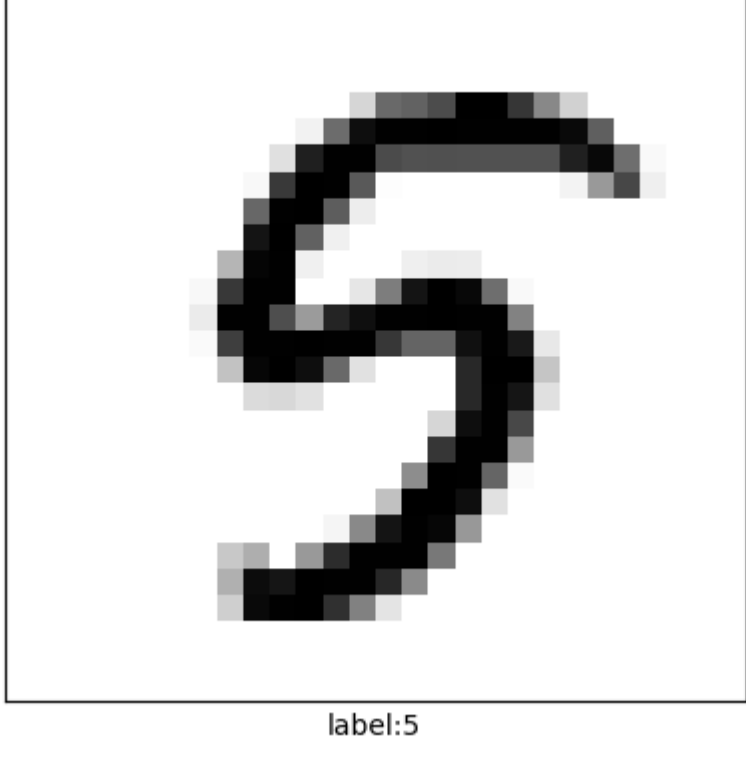
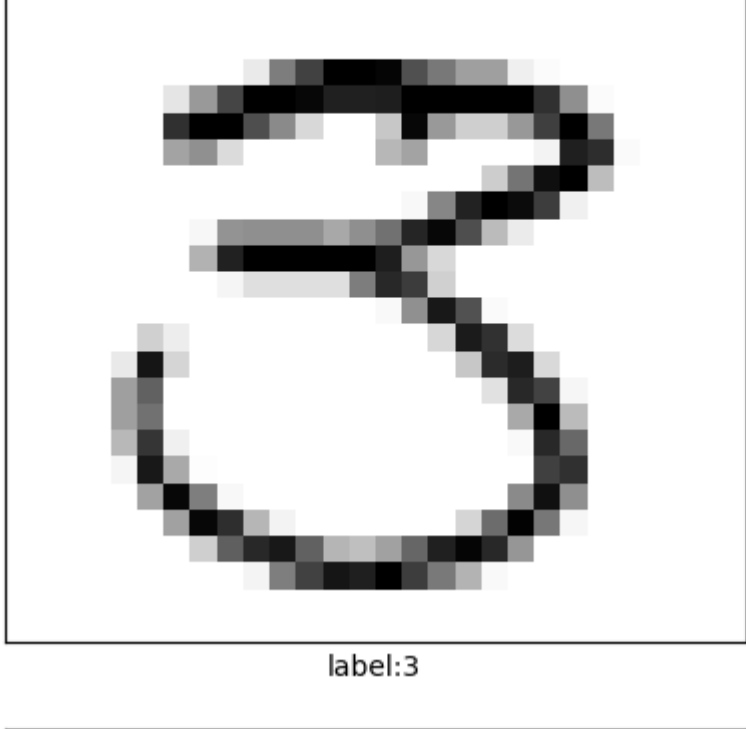
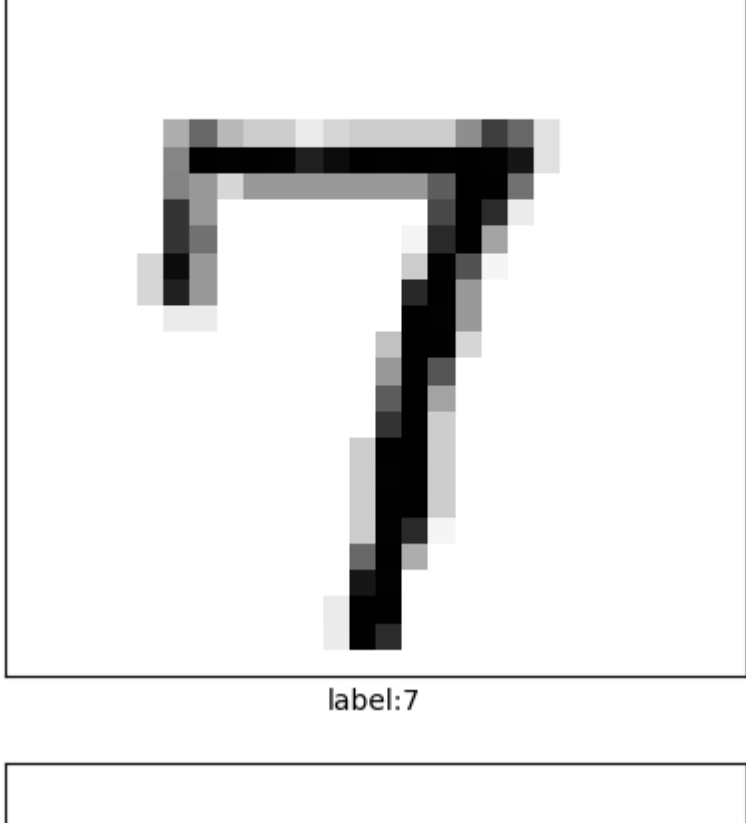
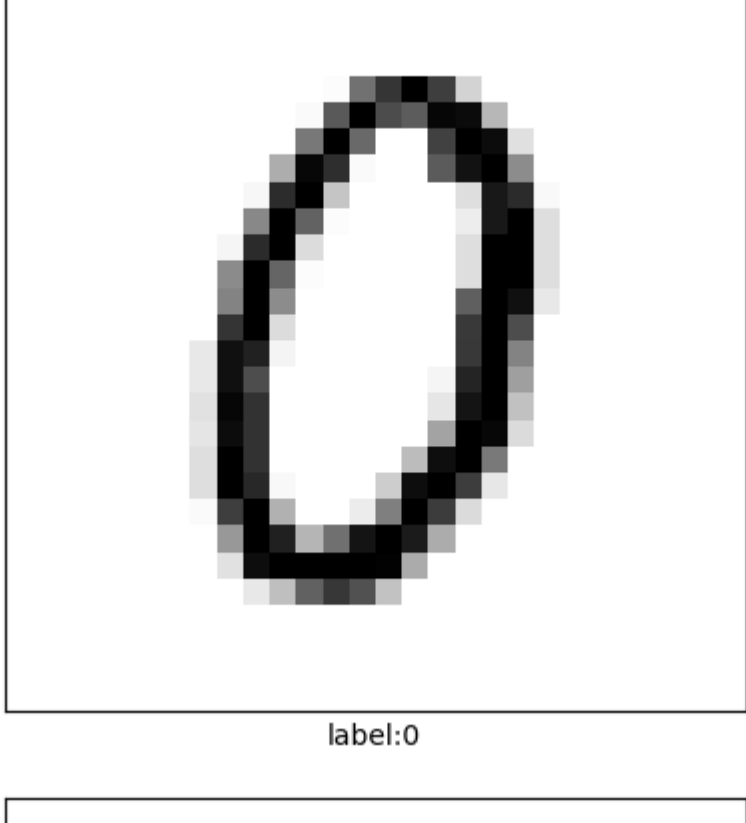
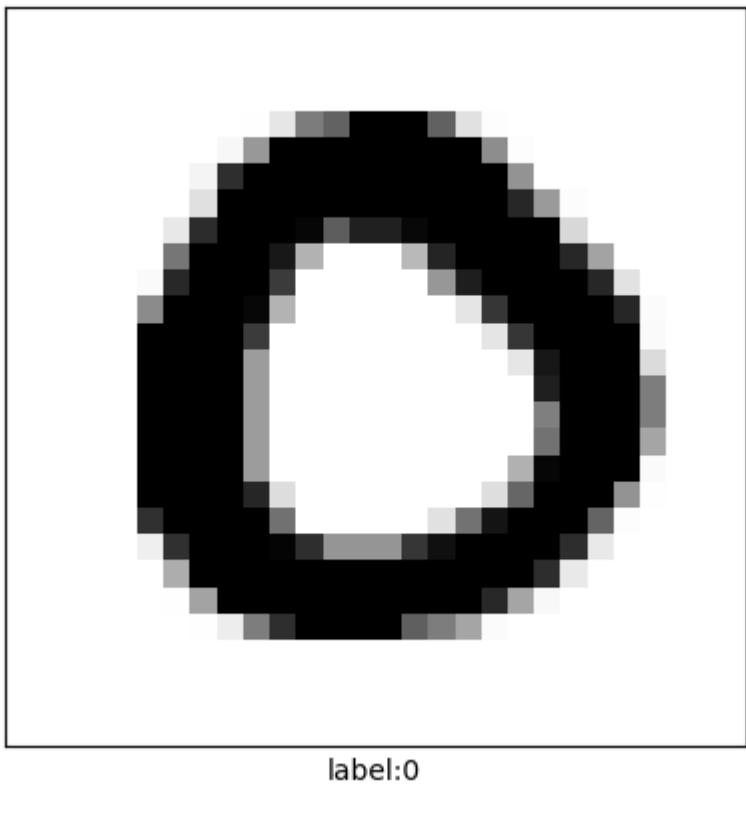
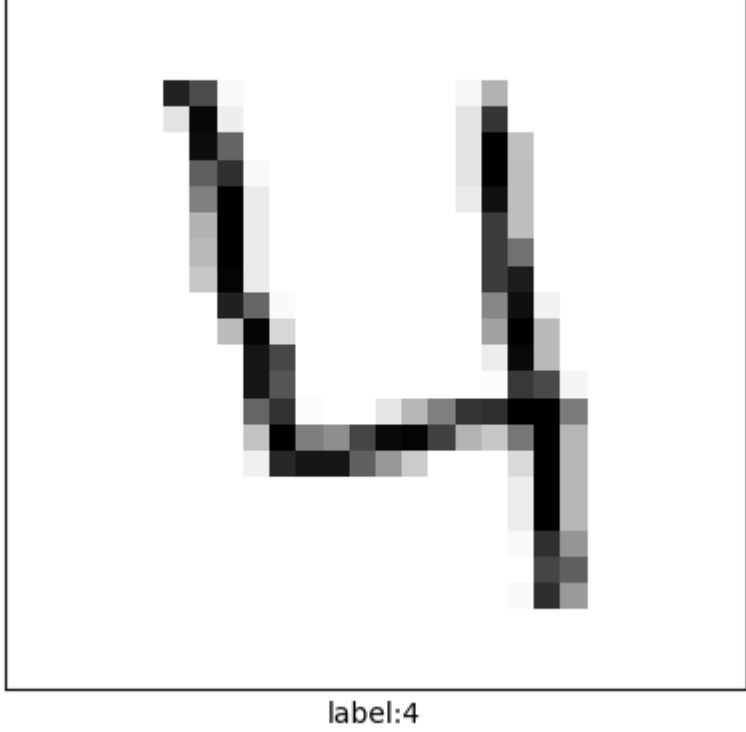
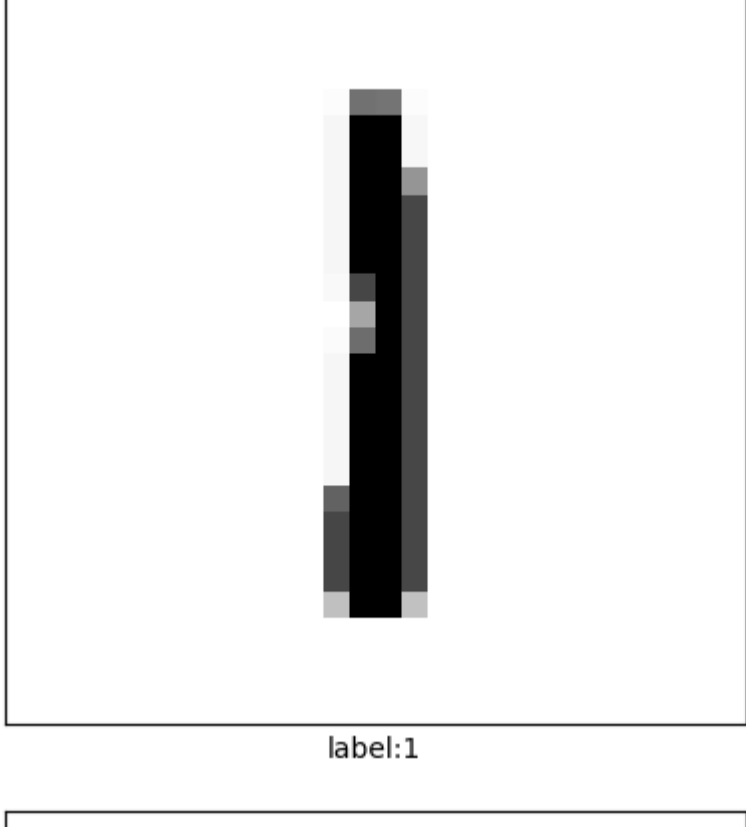
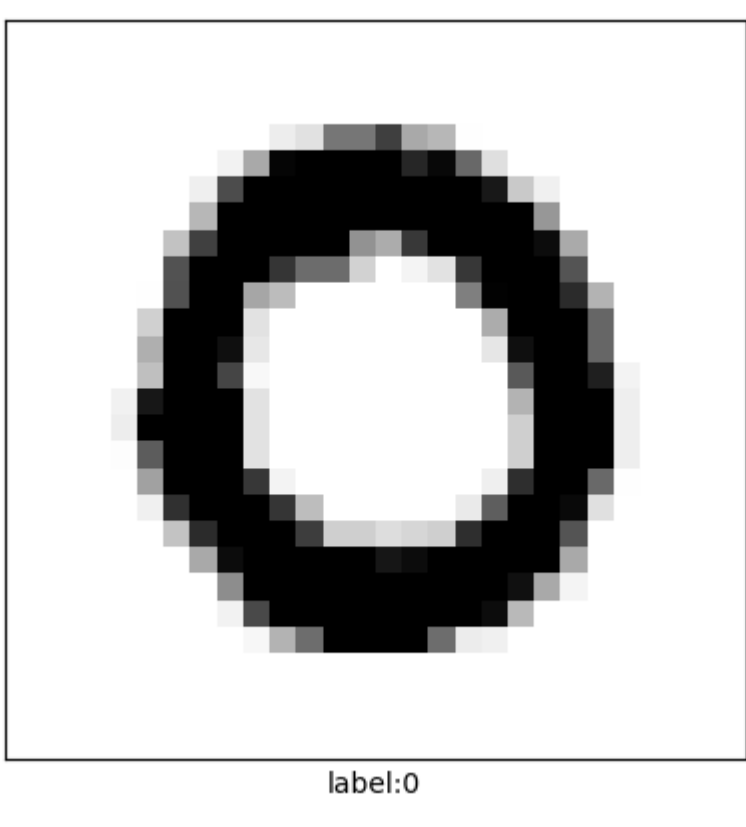
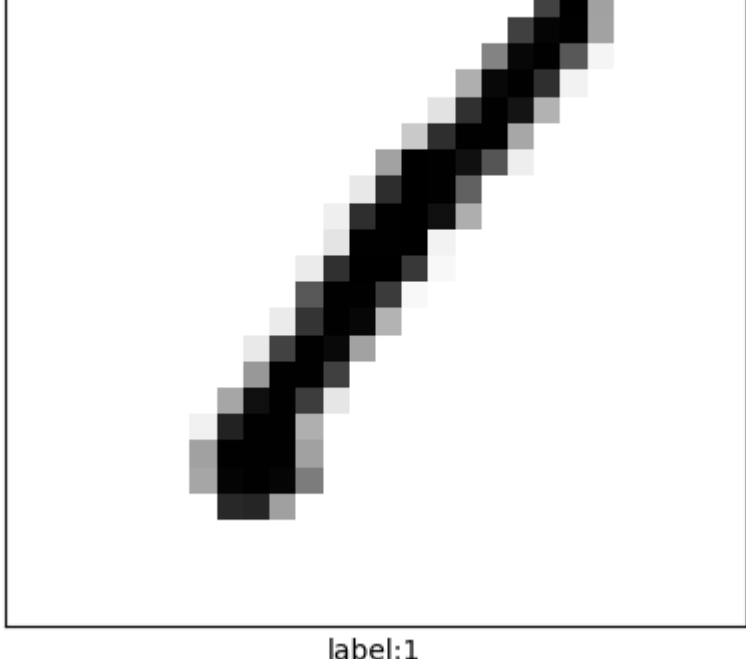
In [17]: def print_image(index):
some_digit = train.iloc[index,1:].values
some_digit_img = some_digit.reshape(28,28)
plt.imshow(some_digit_img, 'binary')

In [20]: print_image(2000)
```



```
In [24]: x = train.drop('label', axis=1)
print("features shape: ", x.shape)
y = train.label
print("Target shape: ", y.shape)
features shape: (42000, 784)
Target shape: (42000,)

In [25]: for i in range(10):
plt.subplot(5,4, i+1)
plt.xticks([])
plt.yticks([])
print_image(i)
plt.xlabel('label:{}'.format(y[i]))
plt.show()
```



```
In [32]: from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, random_state=42, test_size=0.3)
```

## Decision Tree

```
In [32]: dt = DecisionTreeClassifier()
dt = dt.fit(X_train, y_train)

In [33]: y_pred_test_dtr = dt.predict(X_test)
y_pred_train_dtr = dt.predict(X_train)

In [35]: y_pred_test_dtr

Out[35]: array([6, 1, 0, ..., 5, 5, 0], dtype=int64)

In [36]: y_pred_train_dtr

Out[36]: array([4, 0, 0, ..., 2, 6, 0], dtype=int64)

In [37]: print ("Accuracy on Training Data", dt.score(X_train, y_train))
print ("Accuracy on Test Data", dt.score(X_test, y_test))
print("Confusion Matrix: ")
print(confusion_matrix(y_test, y_pred_test_dtr))

print("\nClassification Report:")
print(classification_report(y_test, y_pred_test_dtr))

Accuracy on Training Data 1.0
Accuracy on Test Data 0.840873858783651
Confusion Matrix:
[[1079  8 22 10 11 19 23 7 16  0]
 [ 14 23 1052 52 23 19 22 36 33 28]
 [ 10 19 42 1071 17 82 18 22 45 37]
 [  5 19 13 22 1037 10 19 16 31 50]
 [ 12 13 14 74 19 849 33 15 41 19]
 [ 21  2 23 10 30 25 1108  3 27  7]
 [ 10 11 27 22 22 11 1 1211  9 35]
 [ 13 20 44 51 35 48 25 5 935 25]
 [ 14  2 11 30 58 22  3 41 47 1003]]

Classification Report:
              precision    recall  f1-score   support

 0       0.92       0.90       0.91       1208
 1       0.92       0.94       0.93       1389
 2       0.93       0.91       0.92       1254
 3       0.79       0.79       0.79       1355
 4       0.83       0.85       0.84       1222
 5       0.78       0.78       0.78       1085
 6       0.89       0.88       0.88       1256
 7       0.89       0.89       0.89       1359
 8       0.77       0.77       0.77       1209
 9       0.82       0.81       0.82       1231

 accuracy          0.84
 macro avg         0.84
 weighted avg      0.85
```

## RandomForest

```
In [56]: rf = RandomForestClassifier()
rf.fit(X_train, y_train)

Out[56]: RandomForestClassifier()

In [39]: train_pred_rf = rf.predict(X_train)
pred_rf = rf.predict(X_test)

In [40]: print("Training Accuracy: ", accuracy_score(y_train, train_pred_rf))
print("Test Accuracy: ", accuracy_score(y_test, pred_rf))
print("Confusion Matrix: ")
print(confusion_matrix(y_test, pred_rf))

print("\nClassification Report:")
print(classification_report(y_test, pred_rf))

Training Accuracy: 1.0
Test Accuracy: 0.9011111111111111
Confusion Matrix:
[[1187  0  2  1  2  1  5  0  2  6]
 [  5 5 1235  7 11  2  5 13 10  1]
 [  3  1 17 1270  2 16  4 15 17 10]
 [  2  0  2  0 1102  1  6  1  3 25]
 [  0  1  3 14  1 1033 12  3  5  7]
 [  9  2  1  0  2  6 1231  0  6  0]
 [  1  6 18  1  6  0  0 1300  2 25]
 [  2  9  7 12  6  6  2 1145 12]
 [  8  4  3 23 14  4  2 11  6 1156]]

Classification Report:
              precision    recall  f1-score   support

 0       0.97       0.99       0.98       1208
 1       0.96       0.99       0.98       1389
 2       0.96       0.95       0.95       1254
 3       0.95       0.94       0.95       1355
 4       0.96       0.97       0.97       1222
 5       0.97       0.95       0.96       1085
 6       0.96       0.98       0.97       1256
 7       0.97       0.96       0.96       1359
 8       0.96       0.95       0.95       1209
 9       0.93       0.94       0.94       1231

 accuracy          0.96
 macro avg         0.96
 weighted avg      0.96
```

```
In [41]: test.head()

Out[41]:
```

|   | pixel0 | pixel1 | pixel2 | pixel3 | pixel4 | pixel5 | pixel6 | pixel7 | pixel8 | pixel9 | ... | pixel774 | pixel775 | pixel776 | pixel777 | pixel778 | pixel779 | pixel780 | pixel781 | pixel782 | pixel783 |
|---|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|-----|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| 0 | 0      | 0      | 0      | 0      | 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        | 0        | 0        | 0        |
| 2 | 0      | 0      | 0      | 0      | 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        | 0        | 0        | 0        |
| 4 | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | ... | 0        | 0        | 0        | 0        | 0        | 0        | 0        | 0        | 0        | 0        |

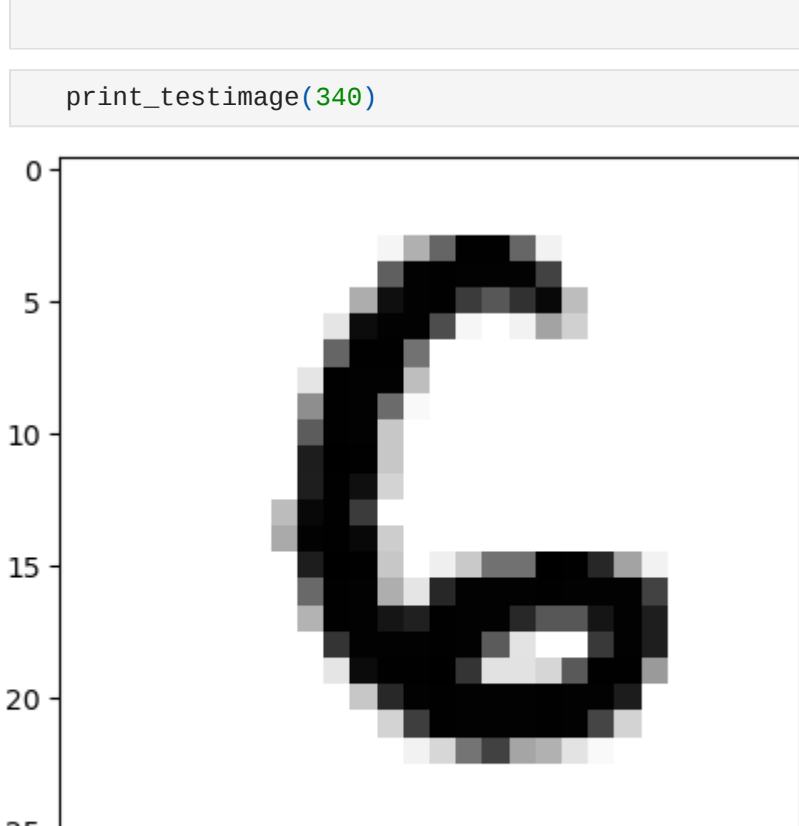
5 rows × 784 columns

```
In [42]: test.shape

Out[42]: (28000, 784)

In [45]: def print_testimage(index):
some_digit = test.iloc[index].values
some_digit_img = some_digit.reshape(28,28)
plt.imshow(some_digit_img, 'binary')

In [46]: print_testimage(240)
```



```
In [48]: testdata2 = dt.predict(test)
testdata2

Out[48]: array([2, 0, 9, ..., 3, 9, 2], dtype=int64)

In [50]: testdata3 = rf.predict(test)
testdata3

Out[50]: array([2, 0, 0, ..., 3, 9, 2], dtype=int64)
```

## Random Prediction

```
print_testimage(1960)

In [53]: # DecisionTree
testdata2[1897]

Out[53]: 3

In [55]: # RandomForest
testdata3[1897]

Out[55]: 3
```

...RandomForest is best at predicting Handwritten digits ie it got accuracy of 96%

...Decision Tree it got less accuracy then RandomForest