In [100…
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
```

In [3]:
```
train = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")
```

In [8]:
```
train.head()
```

Out[8]:

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel77
0	1	0	0	0	0	0	0	0	0	0	...	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	
2	1	0	0	0	0	0	0	0	0	0	...	0	
3	4	0	0	0	0	0	0	0	0	0	...	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	

5 rows × 785 columns

In [5]:
```
train.describe()
```

Out[5]:

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel
count	42000.000000	42000.0	42000.0	42000.0	42000.0	42000.0	42000.0	42000.0	42000.0	42000.
mean	4.456643	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
std	2.887730	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
min	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
25%	2.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
50%	4.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
75%	7.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
max	9.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.

8 rows × 785 columns

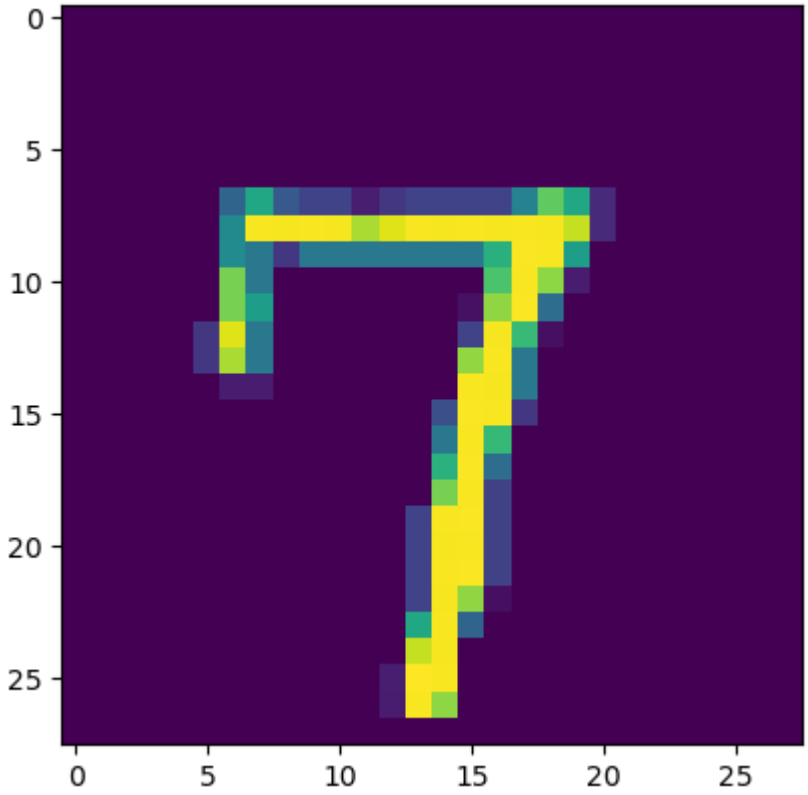
In [10]:
```
train.isna().sum()
```

Out[10]:
```
label       0
pixel0      0
pixel1      0
pixel2      0
pixel3      0
           ..
pixel779    0
pixel780    0
pixel781    0
pixel782    0
pixel783    0
Length: 785, dtype: int64
```

In [33]:
```
number = np.array(train.iloc[6,1:]).reshape(28,28)
```

In [34]:
```
plt.imshow(number)
```

Out[34]:
```
<matplotlib.image.AxesImage at 0x2e75805da90>
```

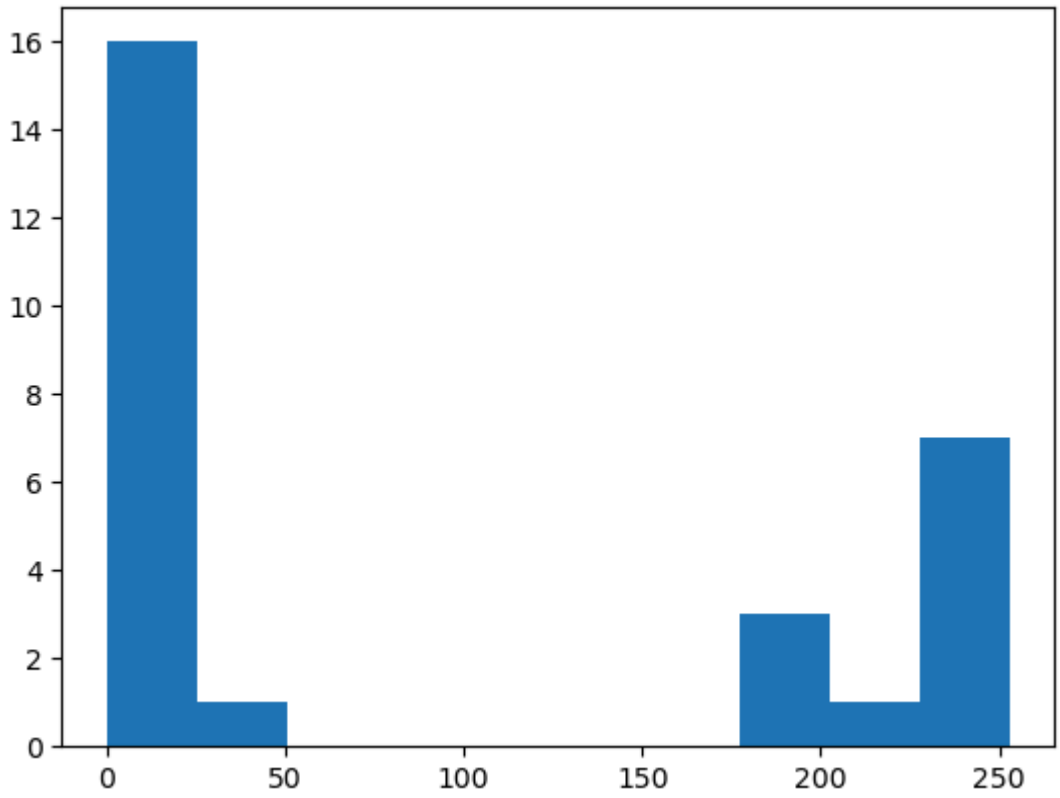


In [35]:
```
number2 = np.array(train.iloc[10,1:]).reshape(28,28)
```

In [43]:
```
number2[23]
```

Out[43]:
```
array([  0,   0,   0,   0,   0,   7, 186, 252, 227, 184, 191, 252, 252,
       252, 252, 253, 240,  50,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0], dtype=int64)
```

In [46]:
```
plt.hist(number2[23])
plt.show()
```



In [47]:
```
x = np.array(train)
y = np.array(test)
```

In [48]:
```
x
```

Out[48]:
```
array([[1, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [1, 0, 0, ..., 0, 0, 0],
       ...,
       [7, 0, 0, ..., 0, 0, 0],
       [6, 0, 0, ..., 0, 0, 0],
       [9, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

In [49]:
```
y
```

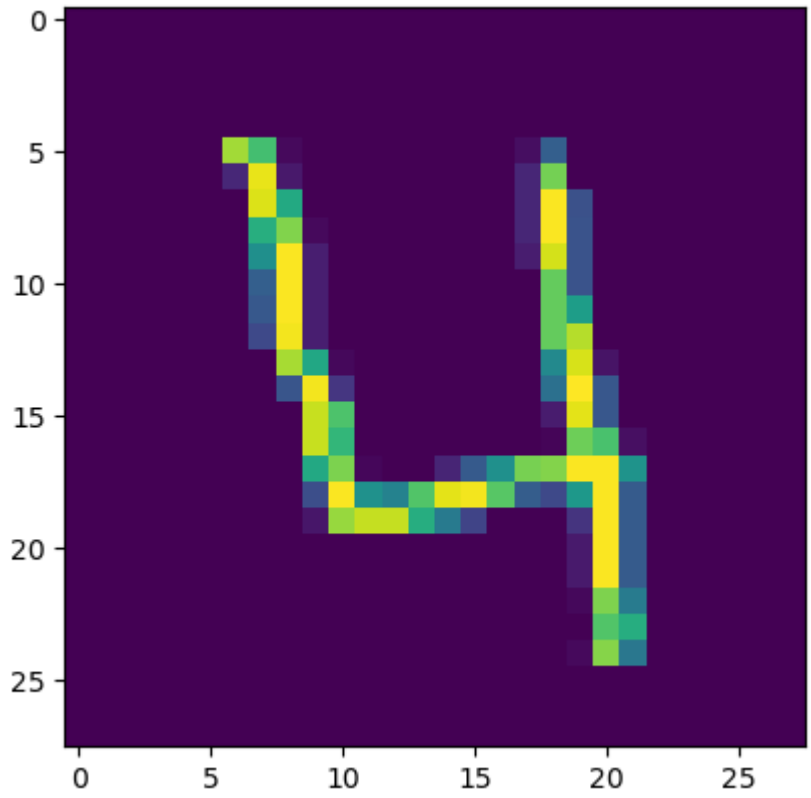
Out[49]:
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

In [77]:
```
x_train = x[:,1:]
y_train = x[:,0]
x_test = y
```

In [78]:
```
d = x_train[3].reshape(28,28)
```

In [79]:
```
plt.imshow(d)
```

Out[79]:
```
<matplotlib.image.AxesImage at 0x2e736601910>
```



In [80]:
```
y
```

Out[80]:
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

In [81]:
```
model = KNeighborsClassifier(n_neighbors=10)
```

In [82]:
```
model
```

Out[82]:
```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=10)
```

In [83]:
```
model.fit(x_train , y_train)
```

Out[83]:
```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=10)
```

In [98]:
```
model.predict(x_test[56].reshape(1,-1))
```

Out[98]:
```
array([1], dtype=int64)
```

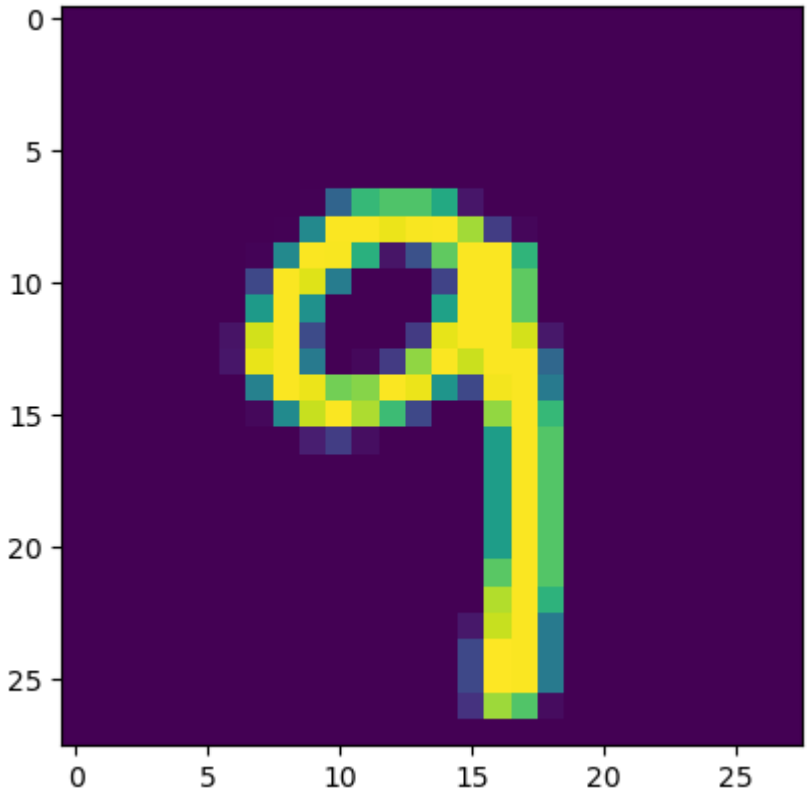
In [92]:
```
int(model.predict(x_test[400].reshape(1,-1)))
```

Out[92]:
```
9
```

In [93]:
```
number_arry = np.array(test.iloc[400,:]).reshape(28,28)
```

In [94]:
```
plt.imshow(number_arry)
```

Out[94]:
```
<matplotlib.image.AxesImage at 0x2e748decc10>
```



In [ ]:
```
```

In [102…
```
logistic = LogisticRegression()
```

In [103…
```
logistic.fit(x_train,y_train)
```

```
C:\Users\godde\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:460:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

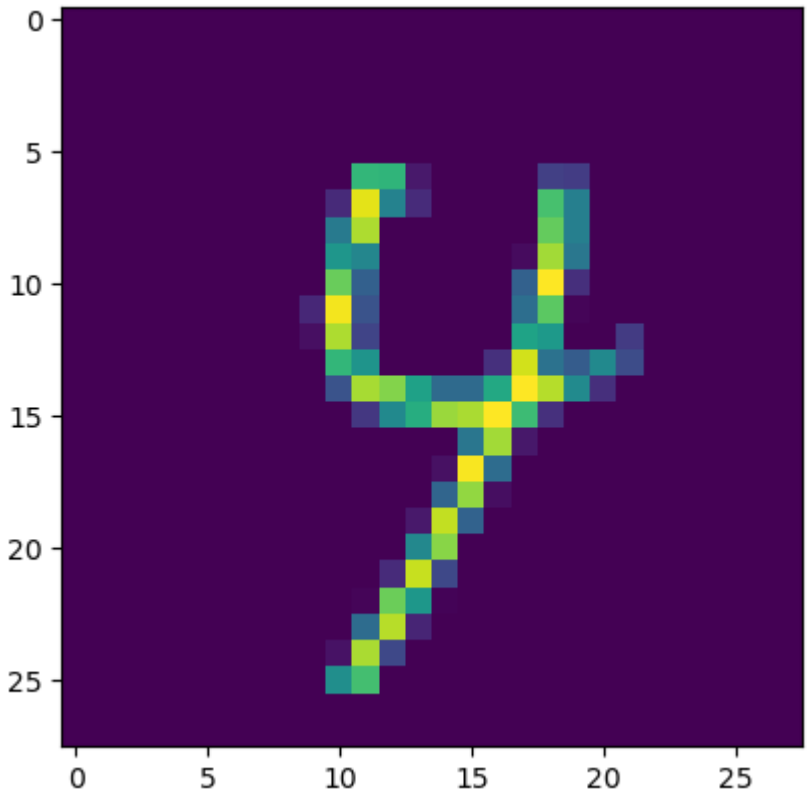
Out[103]:
```
▾ LogisticRegression

LogisticRegression()
```

In [104…
```
logistic.predict(x_test[60].reshape(1,-1))
```

Out[104]:
```
array([4], dtype=int64)
```

In [106…
```
k = np.array(x_test[60,:]).reshape(28,28)
plt.imshow(k)
```

Out[106]:
```
<matplotlib.image.AxesImage at 0x2e749049d90>
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: