MNIST Handwritten Digits Classification using Stochastic Gradient Descent (SGD)

In [2]:

```
import numpy as np
import seaborn as sns
import matplotlib as mpl
import matplotlib.pyplot as plt
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import confusion_matrix
from sklearn import metrics as ms
from sklearn.datasets import fetch_openml
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
```

In [3]:

```
mnist_digits = fetch_openml('mnist_784', version=1)
mnist_digits.keys()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/datasets/_openml.py:968: FutureWarning: T
he default value of `parser` will change from `'liac-arff'` to `'auto'` in 1.4. You can s
et `parser='auto'` to silence this warning. Therefore, an `ImportError` will be raised fr
om 1.4 if the dataset is dense and pandas is not installed. Note that the pandas parser m
ay return different data types. See the Notes Section in fetch_openml's API doc for detai
ls.
  warn(
```

Out[3]:

```
dict_keys(['data', 'target', 'frame', 'categories', 'feature_names', 'target_names', 'DES
CR', 'details', 'url'])
```

In [4]:

```
x = mnist_digits["data"]
y = mnist_digits["target"]
```

In [5]:

```
x.shape , y.shape
```

Out[5]:

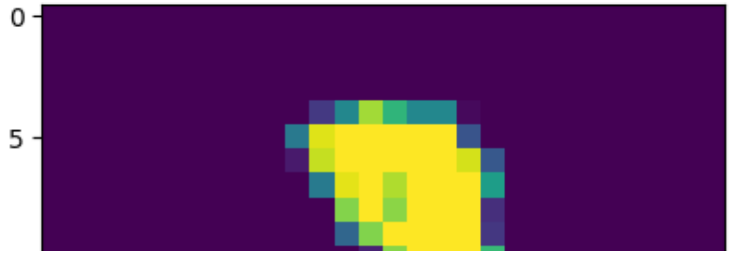
```
((70000, 784), (70000,))
```

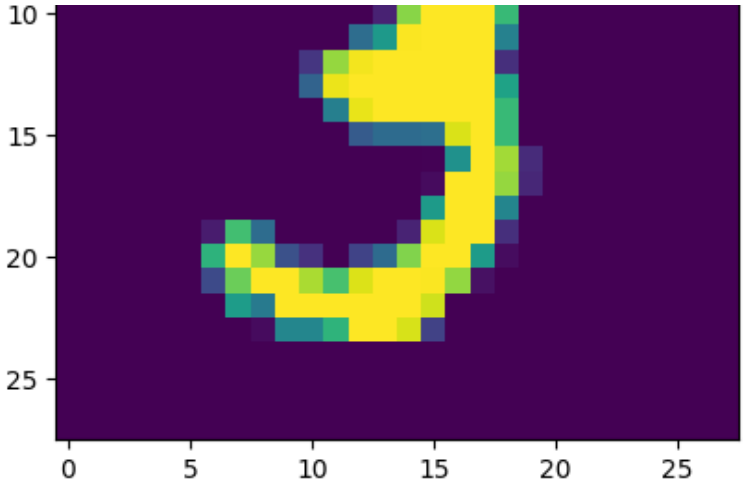
In [10]:

```
sample_a = x.loc[[10]].to_numpy()
sample_b = x.loc[[20]].to_numpy()

sample_a_img = sample_a.reshape(28, 28)
sample_b_img = sample_b.reshape(28, 28)

plt.imshow(sample_a_img)
```

Out[10]:

```
<matplotlib.image.AxesImage at 0x7b34ae700730>
```



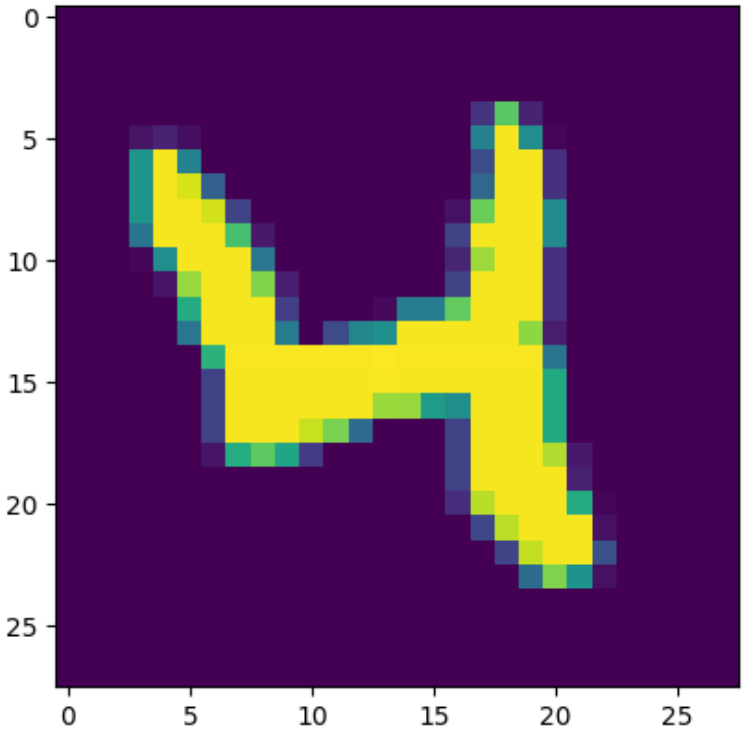


In [11]:

```
plt.imshow(sample_b_img)
```

Out[11]:

```
<matplotlib.image.AxesImage at 0x7b34ae749870>
```



In [12]:

```
x_train, x_test, y_train, y_test = x[:60000], x[60000:], y[:60000], y[60000:]
```

In [13]:

```
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

Out[13]:

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

In [14]:

```
model = SGDClassifier(random_state=42)
model.fit(x_train, y_train)
```

Out[14]:

```
▾ SGDClassifier
SGDClassifier(random_state=42)
```

In [15]:

```
y_predict = model.predict(x_test)
accuracy_score(y_test,y_predict)
```

Out[15]:

```
0.874
```

In [17]:

```
a = cross_val_score(model, x_test, y_test, cv=3, scoring="accuracy")
a
```

Out[17]:

```
array([0.82783443, 0.85448545, 0.8919892 ])
```

In [18]:

```
y_test_pred = cross_val_predict(model, x_test, y_test, cv=3)
c = confusion_matrix(y_test, y_test_pred)
sns.heatmap(c, annot=True)
```

Out[18]:

```
<Axes: >
```