In [20]:

```
import tensorflow as tf
```

In [21]:

```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
```

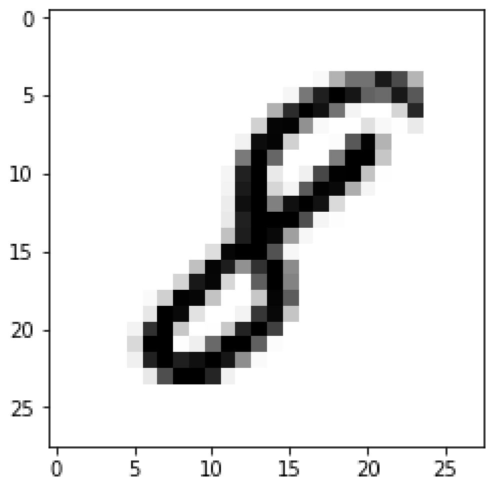
In [22]:

```
import matplotlib.pyplot as plt
image_index = 7777
print(y_train[image_index])
plt.imshow(x_train[image_index], cmap='Greys')
```

```
8
```

Out[22]:

```
<matplotlib.image.AxesImage at 0x1f320e011d0>
```



In [23]:

```
x_train.shape
```

Out[23]:

```
(60000, 28, 28)
```

In [24]:

```
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
input_shape = (28, 28, 1)
# Making sure that the values are float so that we can get decimal points after division
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
# Normalizing the RGB codes by dividing it to the max RGB value.
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print('Number of images in x_train', x_train.shape[0])
print('Number of images in x_test', x_test.shape[0])
```

```
x_train shape: (60000, 28, 28, 1)
Number of images in x_train 60000
Number of images in x_test 10000
```

In [25]:

```
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D
# Creating a Sequential Model and adding the layers
model = Sequential()
model.add(Conv2D(28, kernel_size=(3,3), input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten()) # Flattening the 2D arrays for fully connected layers
model.add(Dense(128, activation=tf.nn.relu))
model.add(Dropout(0.2))
model.add(Dense(10,activation=tf.nn.softmax))
```

In [26]:

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(x=x_train,y=y_train, epochs=5)
```

```
Epoch 1/5
60000/60000 [==============================] - 44s 737us/step - loss: 0.2006
- accuracy: 0.9387
Epoch 2/5
60000/60000 [==============================] - 44s 731us/step - loss: 0.0852
- accuracy: 0.9743
Epoch 3/5
60000/60000 [==============================] - 47s 787us/step - loss: 0.0579
- accuracy: 0.9820
Epoch 4/5
60000/60000 [==============================] - 48s 793us/step - loss: 0.0452
- accuracy: 0.9852
Epoch 5/5
60000/60000 [==============================] - 44s 737us/step - loss: 0.0373
- accuracy: 0.9872
```

Out[26]:

```
<keras.callbacks.callbacks.History at 0x1f321841f60>
```

In [27]:

```

model.evaluate(x_test, y_test)
```

```
10000/10000 [==============================] - 2s 184us/step
```

Out[27]:

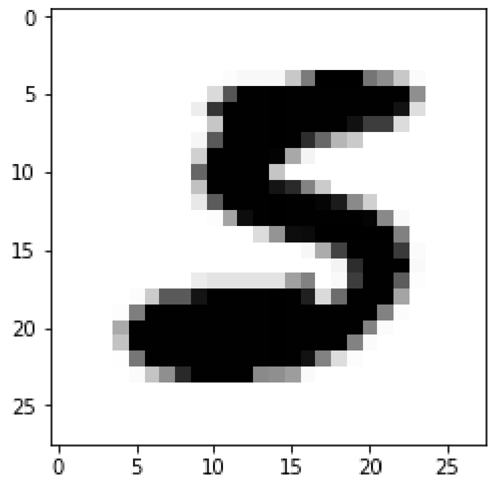
```
[0.056151369899420385, 0.9830999970436096]
```

In [28]:

```
image_index = 7777
plt.imshow(x_test[image_index].reshape(28, 28),cmap='Greys')
```

Out[28]:

```
<matplotlib.image.AxesImage at 0x1f32137b7f0>
```



In [29]:

```
pred = model.predict(x_test[image_index].reshape(1,28,28, 1))
print(pred.argmax())
```

```
5
```