Suraj Shelke

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
import tensorflow as tf
```

```
from tensorflow import keras
```

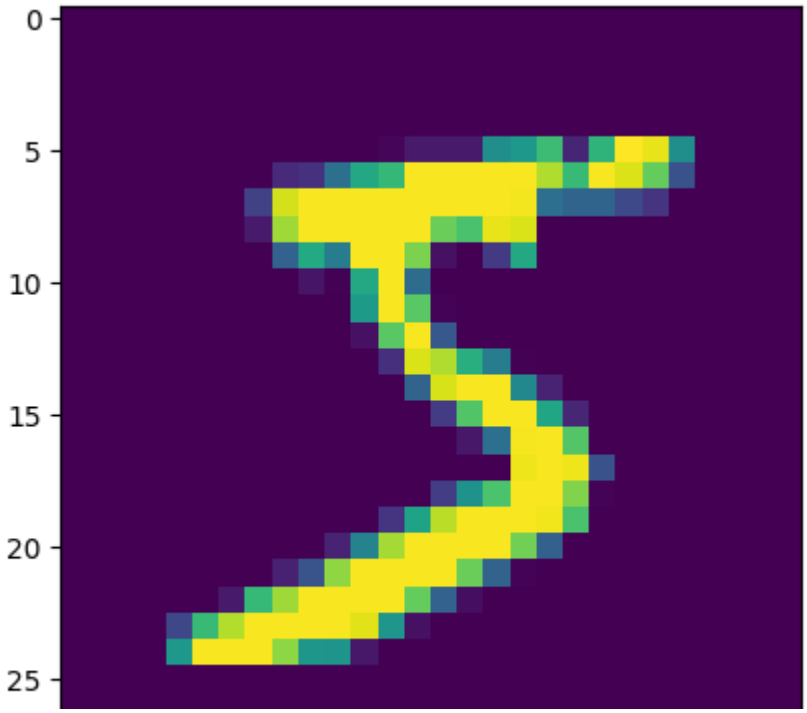
```
digit_mnist = keras.datasets.mnist
(x_train_full, y_train_full), (x_test, y_test) = digit_mnist.load_data()
```

```
x_train_full.shape
```

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

```
plt.imshow(x_train_full[0])
```

```
<matplotlib.image.AxesImage at 0x782694b62ec0>
```




```
x_train_n = x_train_full / 255.
x_test_n = x_test / 255.
```

```
# Load the MNIST dataset.
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

```
# Preprocess the data.
x_train, x_test = x_train / 255.0, x_test / 255.0
```

```
# Build the model.
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])



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

```
Epoch 1/5
1875/1875 [==============================] - 9s 4ms/step - loss: 0.2958 - acc
Epoch 2/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1407 - acc
Epoch 3/5
1875/1875 [==============================] - 7s 4ms/step - loss: 0.1069 - acc
Epoch 4/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0876 - acc
Epoch 5/5
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0738 - acc
<keras.src.callbacks.History at 0x782694bfbfd0>
```

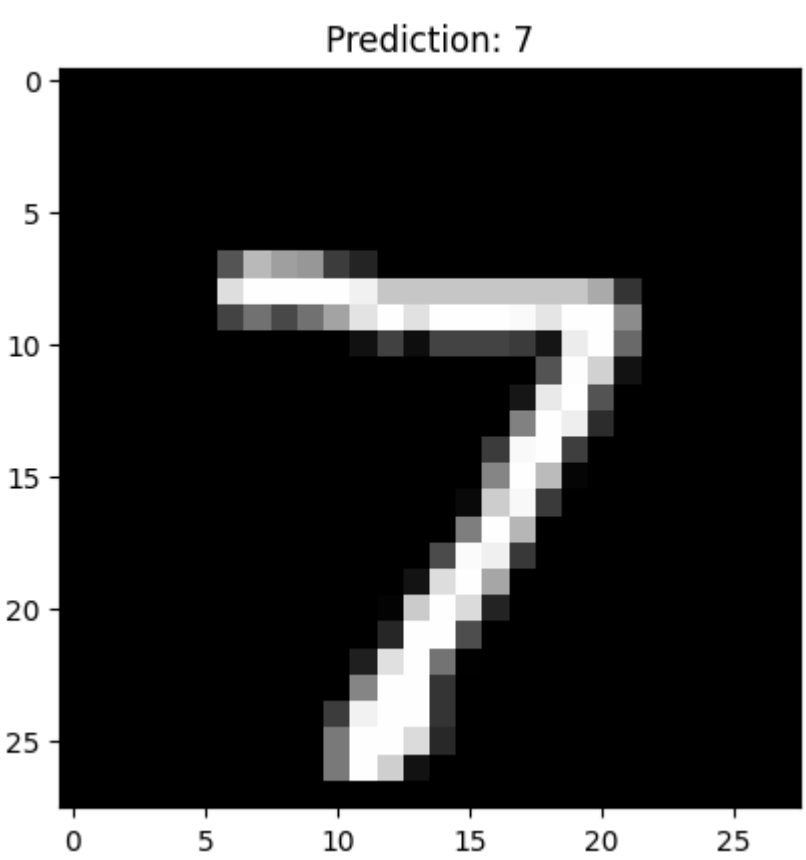
```
# Evaluate the model.
model.evaluate(x_test, y_test)
```

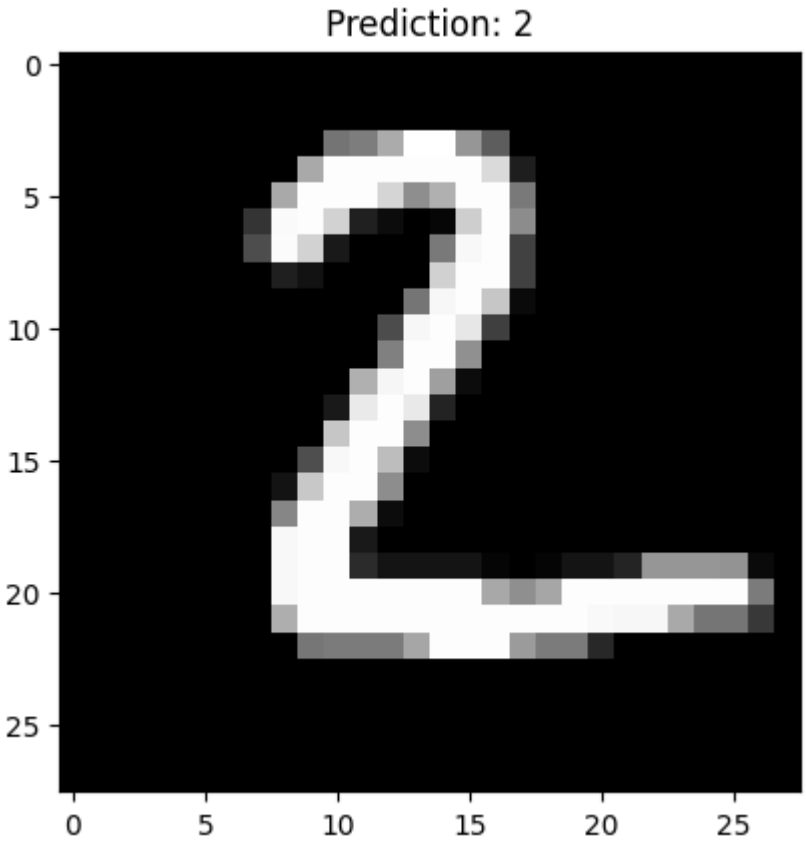
```
313/313 [==============================] - 1s 2ms/step - loss: 0.0745 - accur
[0.07452641427516937, 0.9775999784469604]
```

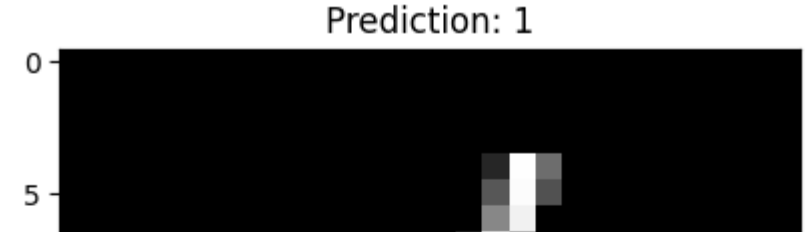
```
# Make predictions.
predictions = model.predict(x_test)
```

```
313/313 [==============================] - 1s 2ms/step
```

```
# Display the results.
for i in range(10):
  plt.imshow(x_test[i], cmap='gray')
  plt.title(f"Prediction: {np.argmax(predictions[i])}")
  plt.show()
```


Prediction: 7
0
5
10
15
20
25
0
5
10
15
20
25


Prediction: 2
0
5
10
15
20
25
0
5
10
15
20
25


Prediction: 1
0
5