```
2.12.0
```

```
fashion_mnist = tf.keras.datasets.fashion_mnist

(train_images, train_labels), (test_images, test_labels) =
fashion_mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-
keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-
keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 [==============================] - 1s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-
keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-
keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 [==============================] - 1s 0us/step
```

```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

```
train_images.shape
```

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

```
len(train_labels)
```

```
60000
```

```
train_labels
```

```
array([9, 0, 0, ..., 3, 0, 5], dtype=uint8)
```

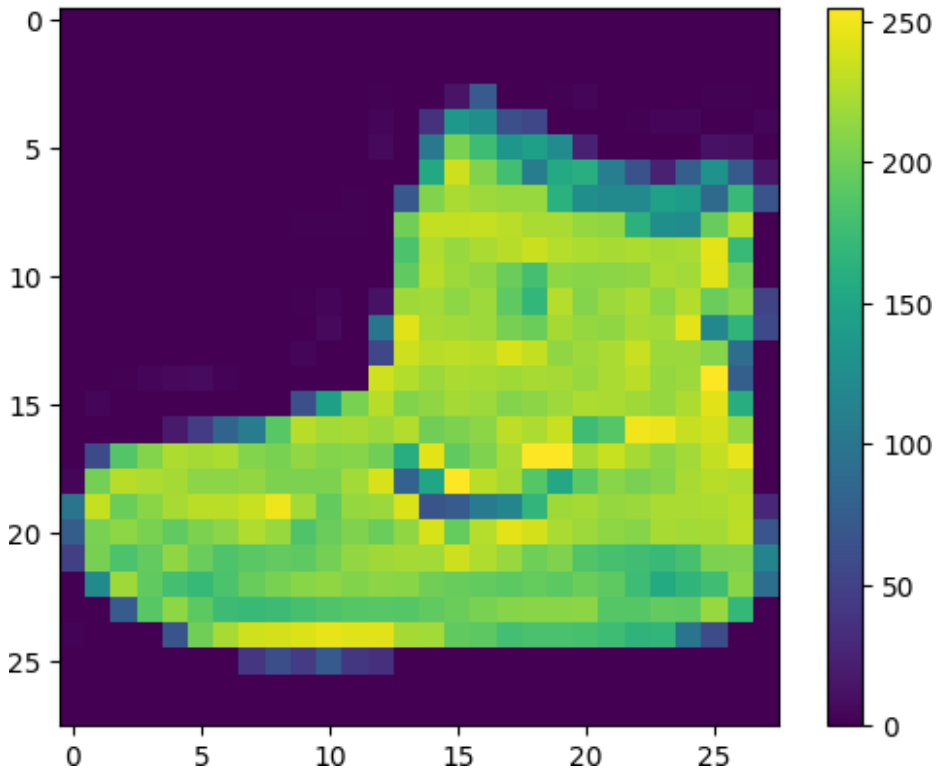
```
test_images.shape
```

```
(10000, 28, 28)
```

```
len(test_labels)
```

```
10000
```

```
plt.figure()
plt.imshow(train_images[0])
plt.colorbar()
plt.grid(False)
plt.show()
```




```
train_images = train_images / 255.0

test_images = test_images / 255.0

plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
```

```
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])
plt.show()
```


```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10)
])
```

```
model.compile(optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

model.fit(train_images, train_labels, epochs=10)

Epoch 1/10
1875/1875 [==============================] - 11s 3ms/step - loss:
0.4998 - accuracy: 0.8235
Epoch 2/10
1875/1875 [==============================] - 6s 3ms/step - loss:
0.3776 - accuracy: 0.8641
Epoch 3/10
1875/1875 [==============================] - 5s 3ms/step - loss:
0.3363 - accuracy: 0.8761
Epoch 4/10
1875/1875 [==============================] - 6s 3ms/step - loss:
0.3122 - accuracy: 0.8859
Epoch 5/10
1875/1875 [==============================] - 5s 3ms/step - loss:
0.2930 - accuracy: 0.8922
Epoch 6/10
1875/1875 [==============================] - 5s 3ms/step - loss:
0.2806 - accuracy: 0.8957
Epoch 7/10
1875/1875 [==============================] - 5s 3ms/step - loss:
0.2663 - accuracy: 0.9007
Epoch 8/10
1875/1875 [==============================] - 5s 3ms/step - loss:
0.2557 - accuracy: 0.9046
Epoch 9/10
1875/1875 [==============================] - 6s 3ms/step - loss:
0.2458 - accuracy: 0.9087
Epoch 10/10
1875/1875 [==============================] - 5s 3ms/step - loss:
0.2369 - accuracy: 0.9113

<keras.callbacks.History at 0x7a7ea01d91b0>

test_loss, test_acc = model.evaluate(test_images, test_labels,
verbose=2)

print('\nTest accuracy:', test_acc)

313/313 - 1s - loss: 0.3358 - accuracy: 0.8806 - 1s/epoch - 4ms/step

Test accuracy: 0.8805999755859375

probability_model = tf.keras.Sequential([model,
                                         tf.keras.layers.Softmax()])
```

```
predictions = probability_model.predict(test_images)
```

```
313/313 [==============================] - 1s 3ms/step
```

```
predictions[0]
```

```
array([6.9260415e-07, 7.0697497e-14, 5.6892759e-12, 1.6533005e-12,
       5.0840594e-09, 5.9169839e-04, 1.0581412e-07, 1.5398487e-02,
       5.5460696e-08, 9.8400903e-01], dtype=float32)
```

```
np.argmax(predictions[0])
```

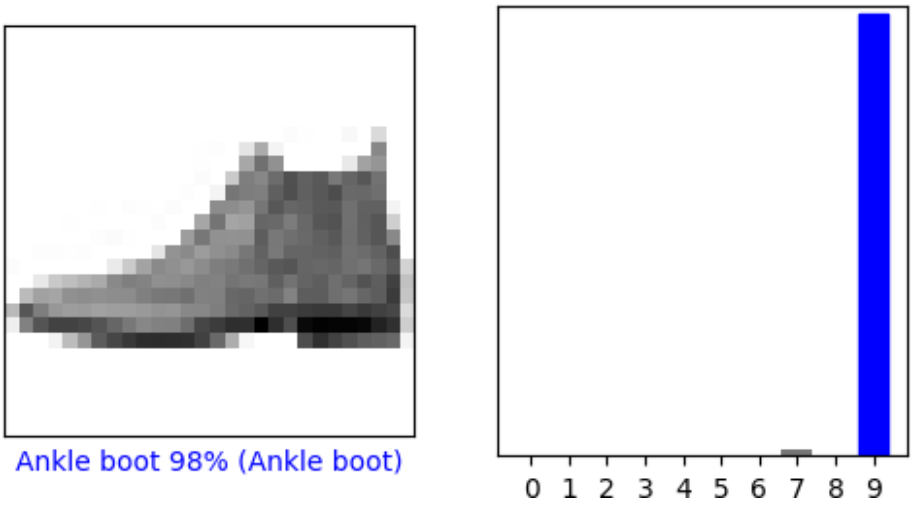
```
9
```

```
test_labels[0]
```

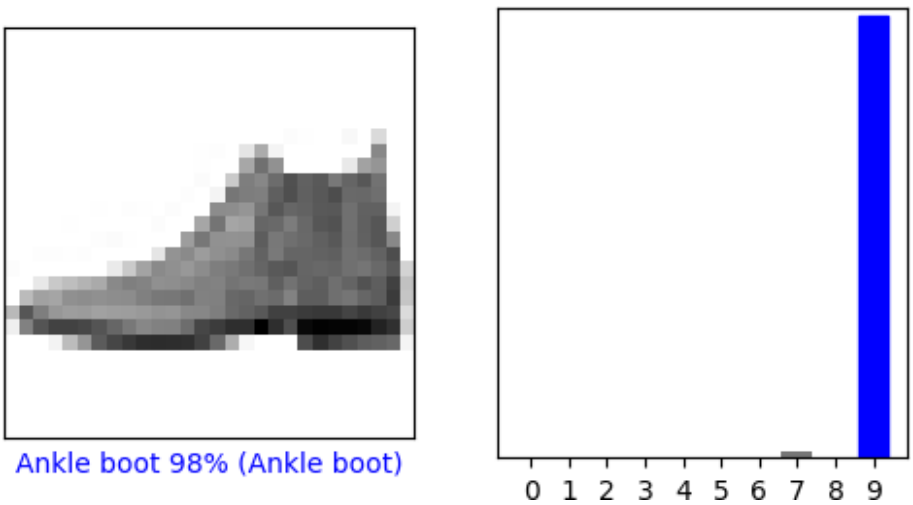
```
9
```

```
def plot_image(i, predictions_array, true_label, img):
  true_label, img = true_label[i], img[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])

  plt.imshow(img, cmap=plt.cm.binary)

  predicted_label = np.argmax(predictions_array)
  if predicted_label == true_label:
    color = 'blue'
  else:
    color = 'red'

  plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                100*np.max(predictions_array),
                                class_names[true_label]),
                                color=color)

def plot_value_array(i, predictions_array, true_label):
  true_label = true_label[i]
  plt.grid(False)
  plt.xticks(range(10))
  plt.yticks([])
  thisplot = plt.bar(range(10), predictions_array, color="#777777")
  plt.ylim([0, 1])
  predicted_label = np.argmax(predictions_array)

  thisplot[predicted_label].set_color('red')
  thisplot[true_label].set_color('blue')

i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
```

```
plot_image(i, predictions[i], test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions[i], test_labels)
plt.show()
```

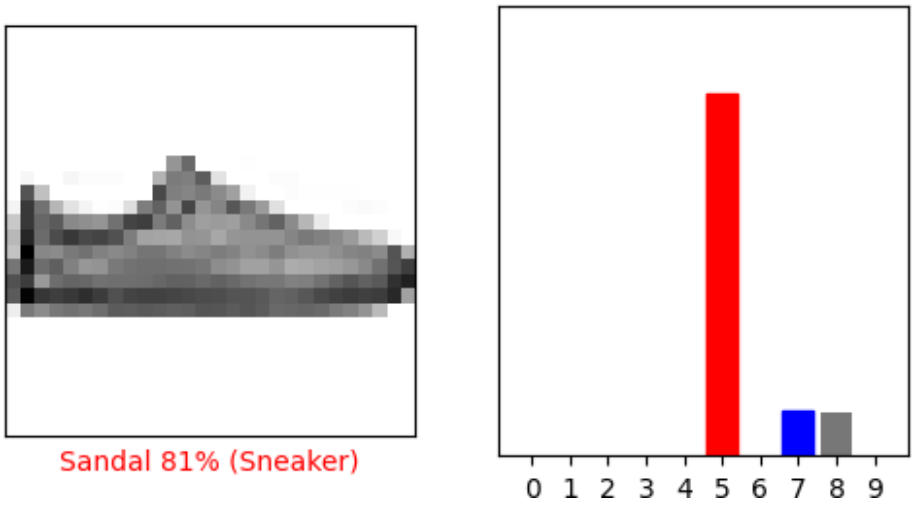



```
i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions[i], test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions[i], test_labels)
plt.show()
```

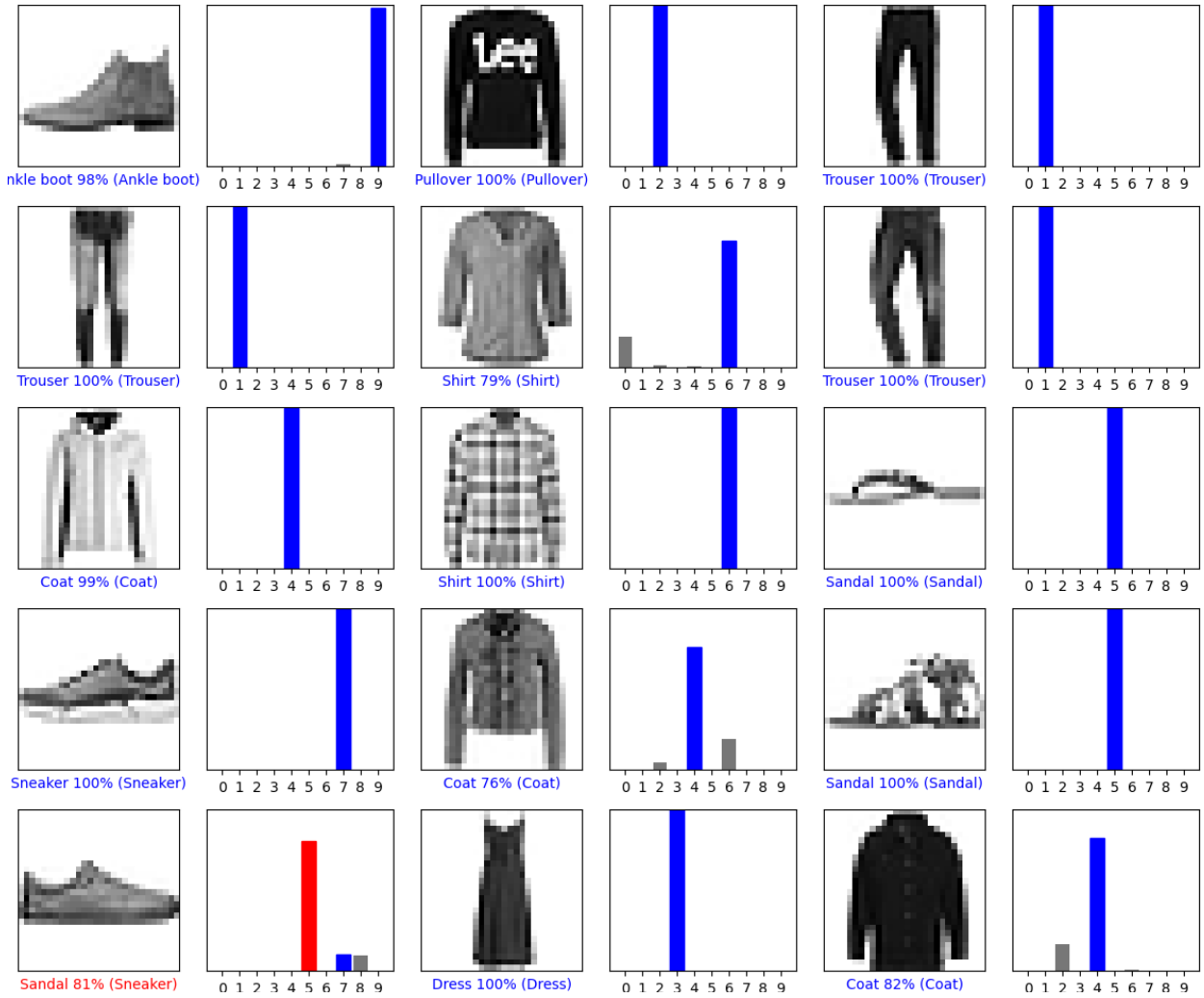



```
i = 12
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
```

```
plot_image(i, predictions[i], test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions[i], test_labels)
plt.show()
```




```
# Plot the first X test images, their predicted labels, and the true labels.
# Color correct predictions in blue and incorrect predictions in red.
num_rows = 5
num_cols = 3
num_images = num_rows*num_cols
plt.figure(figsize=(2*2*num_cols, 2*num_rows))
for i in range(num_images):
  plt.subplot(num_rows, 2*num_cols, 2*i+1)
  plot_image(i, predictions[i], test_labels, test_images)
  plt.subplot(num_rows, 2*num_cols, 2*i+2)
  plot_value_array(i, predictions[i], test_labels)
plt.tight_layout()
plt.show()
```




```
# Grab an image from the test dataset.
img = test_images[1]

print(img.shape)
```

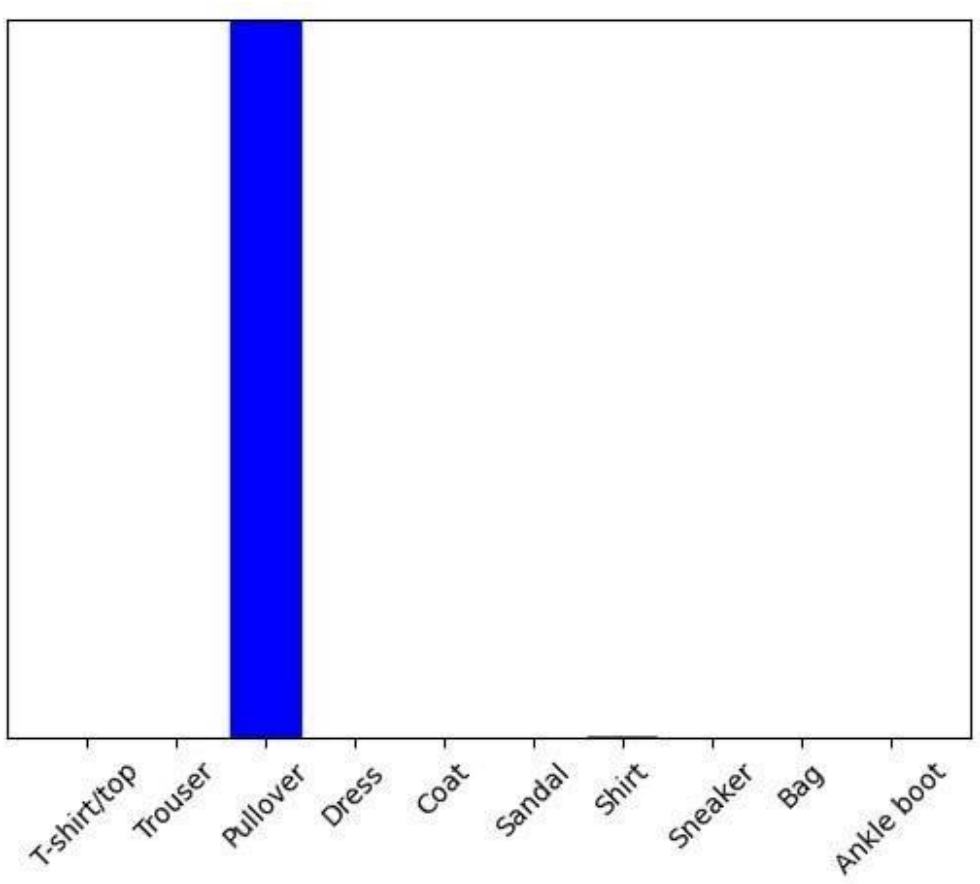
```
(28, 28)
```

```
# Add the image to a batch where it's the only member.
img = (np.expand_dims(img,0))

print(img.shape)
```

```
(1, 28, 28)
```

```
predictions_single = probability_model.predict(img)

print(predictions_single)
```

```
1/1 [==============================] - 0s 24ms/step
[[7.1821800e-05 8.6609013e-16 9.9861825e-01 1.6916174e-10 2.3175428e-
04
  1.1837163e-12 1.0782134e-03 4.4021377e-11 9.3415310e-12 2.7698164e-
09]]

plot_value_array(1, predictions_single[0], test_labels)
_ = plt.xticks(range(10), class_names, rotation=45)
plt.show()
```




```
np.argmax(predictions_single[0])
```

```
2
```

Results and Conclusion - By following the above steps and learning about Keras, TensorFlow (or PyTorch), Google Colab, Kaggle, GitHub, and Git, you have gained essential skills for setting up environments, experimenting with deep learning libraries, utilizing GPU/TPU, collaborating on code projects, and version controlling your codebase. This knowledge empowers you to efficiently work on data science and machine learning projects, and you're now better equipped to dive into more complex tasks and explore advanced techniques.