In [1]:
```
from keras import models, layers
```

In [5]:
```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation= 'relu', input_shape=(28,28,1
)))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(64, (3,3),activation= 'relu'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(64, (3,3), activation= 'relu'))

model.add(layers.Flatten())
model.add(layers.Dense(64, activation= 'relu'))
model.add(layers.Dense(10, activation= 'softmax'))

model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_8 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 3, 3, 64)          36928
_________________________________________________________________
flatten_1 (Flatten)          (None, 576)               0
_________________________________________________________________
dense (Dense)                (None, 64)                36928
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650
=================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
_________________________________________________________________
```

In [6]:
```
from keras.utils import to_categorical
from keras.datasets import mnist
```

In [9]:
```
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

train_images = train_images.reshape((60000, 28, 28, 1))
train_images = train_images.astype('float32') / 255

test_images = test_images.reshape((10000,28, 28, 1))
test_images = test_images.astype('float32') / 255

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)

val_images = train_images[:10000]
x_train = train_images[10000:]
val_labels = train_labels[:10000]
y_train = train_labels[10000:]
```

In [11]:
```
model.compile(optimizer= 'rmsprop',
              loss= 'categorical_crossentropy',
              metrics= ['accuracy'])
history = model.fit(x_train, y_train, epochs= 20, batch_size= 64, validation_d
ata= (val_images, val_labels))
```

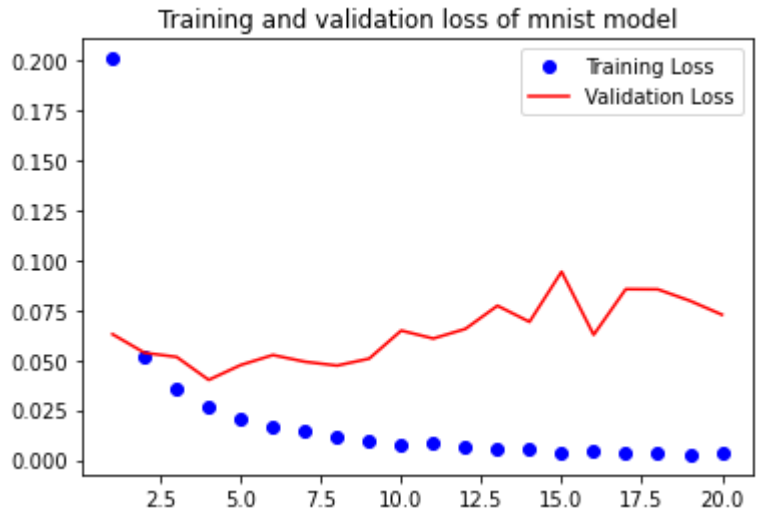
```
Epoch 1/20
782/782 [==============================] - 12s 16ms/step - loss: 0.2010 - acc
uracy: 0.9384 - val_loss: 0.0630 - val_accuracy: 0.9813
Epoch 2/20
782/782 [==============================] - 11s 15ms/step - loss: 0.0518 - acc
uracy: 0.9841 - val_loss: 0.0537 - val_accuracy: 0.9834
Epoch 3/20
782/782 [==============================] - 11s 14ms/step - loss: 0.0356 - acc
uracy: 0.9890 - val_loss: 0.0516 - val_accuracy: 0.9860
Epoch 4/20
782/782 [==============================] - 11s 14ms/step - loss: 0.0270 - acc
uracy: 0.9914 - val_loss: 0.0401 - val_accuracy: 0.9898
Epoch 5/20
782/782 [==============================] - 11s 14ms/step - loss: 0.0209 - acc
uracy: 0.9937 - val_loss: 0.0476 - val_accuracy: 0.9869
Epoch 6/20
782/782 [==============================] - 11s 14ms/step - loss: 0.0164 - acc
uracy: 0.9947 - val_loss: 0.0526 - val_accuracy: 0.9875
Epoch 7/20
782/782 [==============================] - 11s 14ms/step - loss: 0.0142 - acc
uracy: 0.9955 - val_loss: 0.0492 - val_accuracy: 0.9894
Epoch 8/20
782/782 [==============================] - 11s 14ms/step - loss: 0.0112 - acc
uracy: 0.9964 - val_loss: 0.0473 - val_accuracy: 0.9909
Epoch 9/20
782/782 [==============================] - 11s 14ms/step - loss: 0.0092 - acc
uracy: 0.9971 - val_loss: 0.0507 - val_accuracy: 0.9916
Epoch 10/20
782/782 [==============================] - 11s 14ms/step - loss: 0.0077 - acc
uracy: 0.9976 - val_loss: 0.0648 - val_accuracy: 0.9895
Epoch 11/20
782/782 [==============================] - 11s 14ms/step - loss: 0.0081 - acc
uracy: 0.9977 - val_loss: 0.0608 - val_accuracy: 0.9889
Epoch 12/20
782/782 [==============================] - 11s 14ms/step - loss: 0.0069 - acc
uracy: 0.9980 - val_loss: 0.0656 - val_accuracy: 0.9882
Epoch 13/20
782/782 [==============================] - 11s 14ms/step - loss: 0.0052 - acc
uracy: 0.9986 - val_loss: 0.0773 - val_accuracy: 0.9883
Epoch 14/20
782/782 [==============================] - 11s 14ms/step - loss: 0.0054 - acc
uracy: 0.9984 - val_loss: 0.0692 - val_accuracy: 0.9907
Epoch 15/20
782/782 [==============================] - 11s 14ms/step - loss: 0.0035 - acc
uracy: 0.9988 - val_loss: 0.0943 - val_accuracy: 0.9884
Epoch 16/20
782/782 [==============================] - 11s 14ms/step - loss: 0.0041 - acc
uracy: 0.9988 - val_loss: 0.0627 - val_accuracy: 0.9910
Epoch 17/20
782/782 [==============================] - 11s 14ms/step - loss: 0.0034 - acc
uracy: 0.9991 - val_loss: 0.0855 - val_accuracy: 0.9900
Epoch 18/20
782/782 [==============================] - 11s 14ms/step - loss: 0.0040 - acc
uracy: 0.9988 - val_loss: 0.0855 - val_accuracy: 0.9885
Epoch 19/20
782/782 [==============================] - 11s 14ms/step - loss: 0.0027 - acc
uracy: 0.9992 - val_loss: 0.0797 - val_accuracy: 0.9903
```

```
Epoch 20/20
782/782 [==============================] - 11s 14ms/step - loss: 0.0032 - acc
uracy: 0.9991 - val_loss: 0.0728 - val_accuracy: 0.9913
```

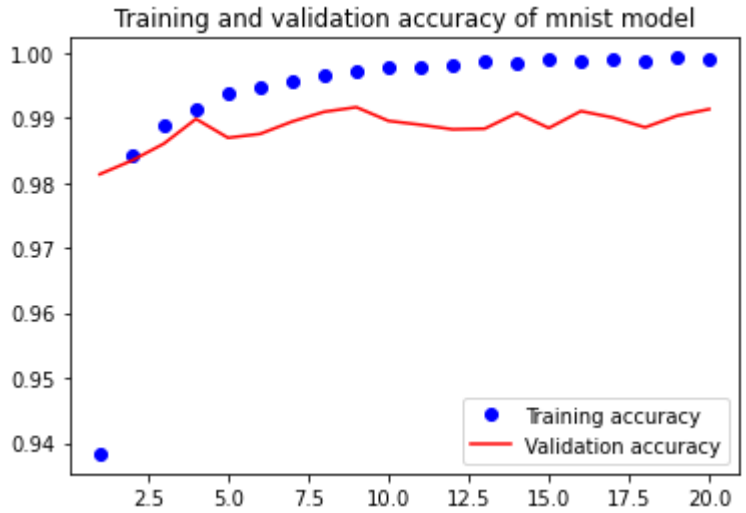
In [36]:
```
import matplotlib.pyplot as plt

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) +1)

plt.plot(epochs, loss, 'bo', label= 'Training Loss')
plt.plot(epochs, val_loss, 'r', label= 'Validation Loss')
plt.title('Training and validation loss of mnist model')
plt.legend()
plt.savefig('Results/6_1/Loss.png');
```



In [37]:
```
plt.plot(epochs, acc, 'bo', label= 'Training accuracy')
plt.plot(epochs, val_acc, 'r', label= 'Validation accuracy')
plt.title('Training and validation accuracy of mnist model')
plt.legend()
plt.savefig('Results/6_1/Accuracy.png');
```



In [19]:
```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation= 'relu', input_shape=(28,28,1
)))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(64, (3,3),activation= 'relu'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(64, (3,3), activation= 'relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation= 'relu'))
model.add(layers.Dense(10, activation= 'softmax'))

model.compile(optimizer= 'rmsprop',
              loss= 'categorical_crossentropy',
              metrics= ['accuracy'])
model.fit(train_images, train_labels, epochs= 9, batch_size= 64)
```

```
Epoch 1/9
938/938 [==============================] - 13s 13ms/step - loss: 0.1837 - acc
uracy: 0.9426
Epoch 2/9
938/938 [==============================] - 12s 13ms/step - loss: 0.0492 - acc
uracy: 0.9849
Epoch 3/9
938/938 [==============================] - 12s 13ms/step - loss: 0.0329 - acc
uracy: 0.9900
Epoch 4/9
938/938 [==============================] - 12s 13ms/step - loss: 0.0248 - acc
uracy: 0.9924
Epoch 5/9
938/938 [==============================] - 12s 13ms/step - loss: 0.0191 - acc
uracy: 0.9942
Epoch 6/9
938/938 [==============================] - 12s 13ms/step - loss: 0.0161 - acc
uracy: 0.9951
Epoch 7/9
938/938 [==============================] - 12s 13ms/step - loss: 0.0127 - acc
uracy: 0.9958
Epoch 8/9
938/938 [==============================] - 12s 13ms/step - loss: 0.0105 - acc
uracy: 0.9968
Epoch 9/9
938/938 [==============================] - 12s 13ms/step - loss: 0.0097 - acc
uracy: 0.9969
```

Out[19]:
```
<tensorflow.python.keras.callbacks.History at 0x7f7d7da764c0>
```

In [38]:
```
model.save('Results/6_1/mode.h5')
```

In [22]:
```
result = model.evaluate(test_images, test_labels)
print(f'The model reult for the test data is loss {result[0]} and accuracy {re
sult[1]}')
```

```
313/313 [==============================] - 1s 4ms/step - loss: 0.0335 - accur
acy: 0.9915
The model reult for the test data is loss 0.03350549191236496 and accuracy 0.
9915000200271606
```

In [40]:
```
import numpy as np
import pandas as pd
preds = model.predict(test_images)
pred_class = np.argmax(preds, axis= 1)
act = np.argmax(test_labels, axis= 1)
pred_df = pd.DataFrame({'Actual':act, 'Predicted':pred_class})
pred_df
```

Out[40]:

	Actual	Predicted
0	7	7
1	2	2
2	1	1
3	0	0
4	4	4
...	...	...
9995	2	2
9996	3	3
9997	4	4
9998	5	5
9999	6	6

10000 rows × 2 columns

In [41]:
```
pred_df.to_csv('Results/6_1/predictions.csv')
```

In [ ]:
```

```