HIDDEN LAYERS	NEURONS	LOSS FUNCTION	OPTIMIZERS	ACTIVATE FUNCTION	DROPOUT	EPOCH(HIGHEST ACCURACY)	VALIDATION ACCURACY	TEST ACCURACY
1	16	MSE	ADAM	TANH		2	0.9115	0.885
1	32	MSE	ADAM	TANH		2	0.9557	0.8799
1	64	MSE	ADAM	TANH	0.9614(dropo	4	0.9287	0.8858
1	128	MSE	ADAM	TANH		2	0.9318	0.8819
2	16	BINARY_CROSSENTROPY	RMSPROP	RELU		2	0.9405	0.8827
2	16	MSE	ADAM	TANH		2	0.9596	0.8762
2	32	MSE	ADAM	TANH		2	0.929	0.8807
2	64	MSE	ADAM	TANH		2	0.9332	0.8775
2	128	MSE	ADAM	TANH		2	0.9555	0.8706
3	16	MSE	ADAM	TANH		2	0.950	0.8746
3	32	MSE	ADAM	TANH		2	0.8696	0.8695
3	64	MSE	ADAM	TANH		2	0.9348	0.8766
3	128	MSE	ADAM	TANH		2	0.935	0.8766

Summary:
1. When used each hidden layers 1,2,3 to execute with neurons 16, 32, 64 we can see that maximum accuracy is attained when hidden layer is 1 and 64 nodes.
2. We have used generalization technique dropout to the one which have maximum accuracy.
3. Comparing the test accuracy values of highest accuracy algorithm one with hyper parameter and without, we can observe that there is not must difference between both the test accuracies.

In [54]:
```
from tensorflow.keras.datasets import imdb
(train_data, train_labels), (test_data, test_labels) = imdb.load_data(
    num_words=10000)
```

In [55]:
```
train_data[0]
```

```
 16,
 38,
 1334,
 88,
 12,
 16,
 283,
 5,
 16,
 4472,
 113,
 103,
 32,
 15,
 16,
 5345,
 19,
 178,
 32]
```

In [56]:
```
train_labels[0]
```

Out[56]: 1

In [57]:
```
max([max(sequence) for sequence in train_data])
```

Out[57]: 9999

In [58]:
```
#Decoding reviews back to text
```

In [59]:
```
word_index = imdb.get_word_index()
reverse_word_index = dict(
    [(value, key) for (key, value) in word_index.items()])
decoded_review = " ".join(
    [reverse_word_index.get(i - 3, "?") for i in train_data[0]])
```

In [60]:
```
#Preparinf data set
```

In [61]:
```
import numpy as np
def vectorize_sequences(sequences, dimension=10000):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        for j in sequence:
            results[i, j] = 1.
    return results
x_train = vectorize_sequences(train_data)
x_test = vectorize_sequences(test_data)
```

In [62]:
```
x_train[0]
```

Out[62]: array([0., 1., 1., ..., 0., 0., 0.])

In [63]:
```
y_train = np.asarray(train_labels).astype("float32")
y_test = np.asarray(test_labels).astype("float32")
```

In [64]:
```
#Buliding Model
```

In [65]:
```
from tensorflow import keras
from tensorflow.keras import layers

model = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
```

In [66]:
```
model.compile(optimizer="adam",
              loss="binary_crossentropy",
              metrics=["accuracy"])
```

In [67]:
```
#Validating your approach
```

In [68]:
```
x_val = x_train[:10000]
partial_x_train = x_train[10000:]
y_val = y_train[:10000]
partial_y_train = y_train[10000:]
```

In [69]:
```
#Training  model
```

In [70]:
```
history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
```

```
Epoch 1/20
30/30 [==============================] - 2s 37ms/step - loss: 0.5723 - a
ccuracy: 0.7423 - val_loss: 0.4117 - val_accuracy: 0.8559
Epoch 2/20
30/30 [==============================] - 1s 20ms/step - loss: 0.3069 - a
ccuracy: 0.8981 - val_loss: 0.2954 - val_accuracy: 0.8897
Epoch 3/20
30/30 [==============================] - 1s 21ms/step - loss: 0.2039 - a
ccuracy: 0.9321 - val_loss: 0.2809 - val_accuracy: 0.8894
Epoch 4/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1508 - a
ccuracy: 0.9517 - val_loss: 0.2881 - val_accuracy: 0.8842
Epoch 5/20
30/30 [==============================] - 1s 21ms/step - loss: 0.1157 - a
ccuracy: 0.9666 - val_loss: 0.2975 - val_accuracy: 0.8834
Epoch 6/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0873 - a
ccuracy: 0.9781 - val_loss: 0.3164 - val_accuracy: 0.8816
Epoch 7/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0670 - a
ccuracy: 0.9857 - val_loss: 0.3394 - val_accuracy: 0.8822
Epoch 8/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0507 - a
ccuracy: 0.9908 - val_loss: 0.3666 - val_accuracy: 0.8782
Epoch 9/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0384 - a
ccuracy: 0.9947 - val_loss: 0.3926 - val_accuracy: 0.8769
Epoch 10/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0292 - a
ccuracy: 0.9973 - val_loss: 0.4190 - val_accuracy: 0.8756
Epoch 11/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0216 - a
ccuracy: 0.9987 - val_loss: 0.4471 - val_accuracy: 0.8732
Epoch 12/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0164 - a
ccuracy: 0.9993 - val_loss: 0.4739 - val_accuracy: 0.8724
Epoch 13/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0128 - a
ccuracy: 0.9995 - val_loss: 0.4960 - val_accuracy: 0.8709
Epoch 14/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0100 - a
ccuracy: 0.9998 - val_loss: 0.5181 - val_accuracy: 0.8708
Epoch 15/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0078 - a
ccuracy: 1.0000 - val_loss: 0.5379 - val_accuracy: 0.8707
Epoch 16/20
30/30 [==============================] - 0s 17ms/step - loss: 0.0063 - a
ccuracy: 1.0000 - val_loss: 0.5573 - val_accuracy: 0.8703
Epoch 17/20
30/30 [==============================] - 0s 17ms/step - loss: 0.0052 - a
```

```
ccuracy: 1.0000 - val_loss: 0.5738 - val_accuracy: 0.8697
Epoch 18/20
30/30 [==============================] - 0s 17ms/step - loss: 0.0044 - a
ccuracy: 1.0000 - val_loss: 0.5906 - val_accuracy: 0.8685
Epoch 19/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0037 - a
ccuracy: 1.0000 - val_loss: 0.6053 - val_accuracy: 0.8684
Epoch 20/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0032 - a
ccuracy: 1.0000 - val_loss: 0.6202 - val_accuracy: 0.8680
```

In [71]:

```
#Compiling the model
```

In [72]:

```
history_dict = history.history
history_dict.keys()
```

Out[72]:

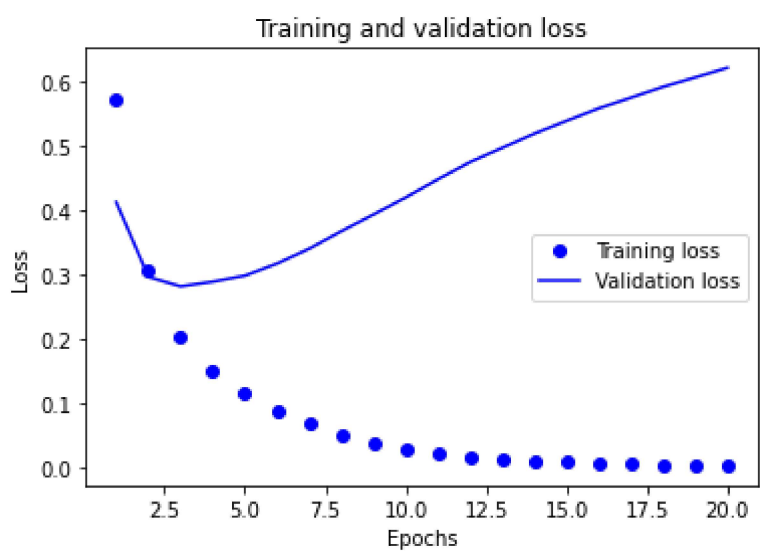
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [73]:

```
#Plotting the training and validation loss
```

In [74]:

```
import matplotlib.pyplot as plt
history_dict = history.history
loss_values = history_dict["loss"]
val_loss_values = history_dict["val_loss"]
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, "bo", label="Training loss")
plt.plot(epochs, val_loss_values, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
```

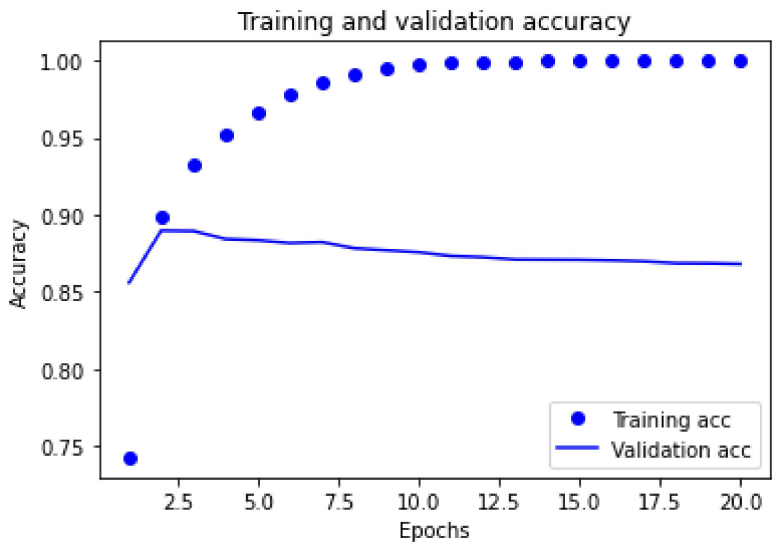


In [75]:

```
#Plotting the training and accuracy
```

In [76]:

```
plt.clf()
acc = history_dict["accuracy"]
val_acc = history_dict["val_accuracy"]
plt.plot(epochs, acc, "bo", label="Training acc")
plt.plot(epochs, val_acc, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```



In [77]:

```
#Retraining a model from scratch
```

In [78]:
```
model = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.fit(x_train, y_train, epochs=4, batch_size=512)
results = model.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 [==============================] - 1s 12ms/step - loss: 0.4489 - accu
racy: 0.8226
Epoch 2/4
49/49 [==============================] - 1s 12ms/step - loss: 0.2586 - accu
racy: 0.9098
Epoch 3/4
49/49 [==============================] - 1s 12ms/step - loss: 0.2018 - accu
racy: 0.9287
Epoch 4/4
49/49 [==============================] - 1s 12ms/step - loss: 0.1692 - accu
racy: 0.9405
782/782 [==============================] - 2s 3ms/step - loss: 0.2959 - acc
uracy: 0.8827
```

In [79]:
```
results
```

Out[79]:
```
[0.29586344957351685, 0.8827199935913086]
```

In [80]:
```
#Using a trained model to generate predictions on new data
```

In [81]:
```
model.predict(x_test)
```

```
782/782 [==============================] - 2s 3ms/step
```

Out[81]:
```
array([[0.17031914],
       [0.9990291 ],
       [0.8739922 ],
       ...,
       [0.08244044],
       [0.05501057],
       [0.4795224 ]], dtype=float32)
```

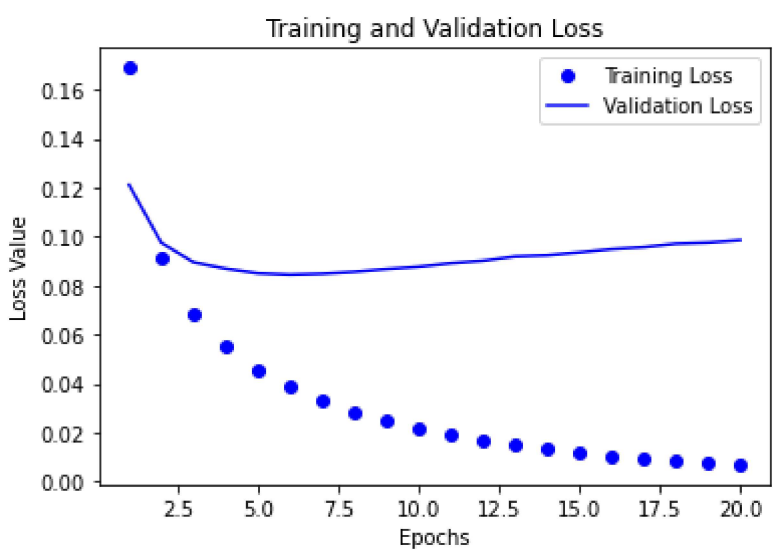
In [82]:

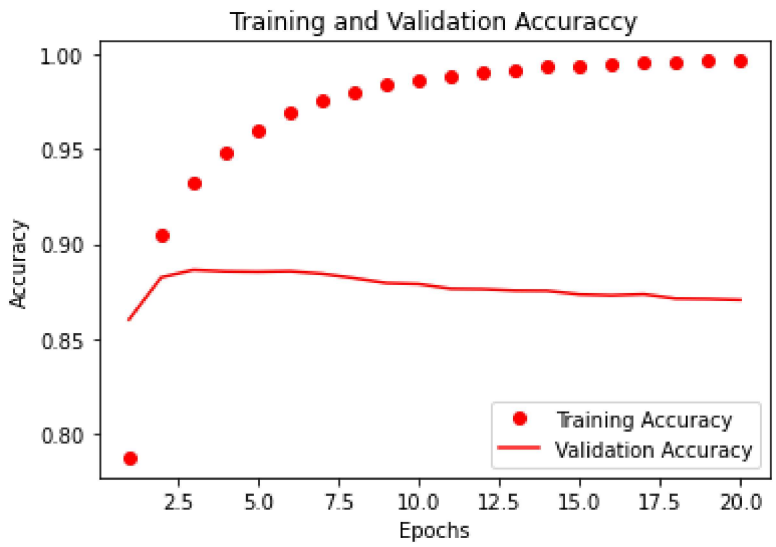
```
#implemented one hideden layer with 16 neurons and mse loss function

from keras import models
from tensorflow.keras import layers

model = models.Sequential()
model.add(layers.Dense(16, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

from keras import optimizers
from keras import losses
from keras import metrics

from tensorflow import keras
from keras import optimizers
from tensorflow.keras import optimizers
from tensorflow.keras import optimizers

model.compile(optimizer='adam',
              loss = losses.mse,
              metrics = [metrics.binary_accuracy])



x_val = x_train[:10000]
partial_x_train = x_train[10000:]

y_val = y_train[:10000]
partial_y_train = y_train[10000:]

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

history_dict = history.history
history_dict.keys()

# Plotting the training and validation loss

import matplotlib.pyplot as plt
%matplotlib inline

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

epochs = range(1, len(loss_values) + 1)

plt.plot(epochs, loss_values, 'bo', label="Training Loss")
plt.plot(epochs, val_loss_values, 'b', label="Validation Loss")
```

```
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss Value')
plt.legend()

plt.show()


# Plotting the training and validation accuracy
# Training and Validation Accuracy

acc_values = history_dict['binary_accuracy']
val_acc_values = history_dict['val_binary_accuracy']

epochs = range(1, len(loss_values) + 1)

plt.plot(epochs, acc_values, 'ro', label="Training Accuracy")
plt.plot(epochs, val_acc_values, 'r', label="Validation Accuracy")

plt.title('Training and Validation Accuraccy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```

```
Epoch 1/20
30/30 [==============================] - 2s 38ms/step - loss: 0.1689 - bina
ry_accuracy: 0.7874 - val_loss: 0.1211 - val_binary_accuracy: 0.8603
Epoch 2/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0916 - bina
ry_accuracy: 0.9045 - val_loss: 0.0975 - val_binary_accuracy: 0.8825
Epoch 3/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0684 - bina
ry_accuracy: 0.9324 - val_loss: 0.0895 - val_binary_accuracy: 0.8864
Epoch 4/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0552 - bina
ry_accuracy: 0.9481 - val_loss: 0.0869 - val_binary_accuracy: 0.8856
Epoch 5/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0456 - bina
ry_accuracy: 0.9603 - val_loss: 0.0850 - val_binary_accuracy: 0.8854
Epoch 6/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0386 - bina
ry_accuracy: 0.9694 - val_loss: 0.0844 - val_binary_accuracy: 0.8857
Epoch 7/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0329 - bina
ry_accuracy: 0.9756 - val_loss: 0.0848 - val_binary_accuracy: 0.8844
Epoch 8/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0284 - bina
ry_accuracy: 0.9800 - val_loss: 0.0856 - val_binary_accuracy: 0.8821
Epoch 9/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0247 - bina
ry_accuracy: 0.9839 - val_loss: 0.0867 - val_binary_accuracy: 0.8795
Epoch 10/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0215 - bina
```

```
ry_accuracy: 0.9867 - val_loss: 0.0877 - val_binary_accuracy: 0.8790
Epoch 11/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0189 - bina
ry_accuracy: 0.9887 - val_loss: 0.0891 - val_binary_accuracy: 0.8765
Epoch 12/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0167 - bina
ry_accuracy: 0.9907 - val_loss: 0.0901 - val_binary_accuracy: 0.8763
Epoch 13/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0148 - bina
ry_accuracy: 0.9922 - val_loss: 0.0919 - val_binary_accuracy: 0.8755
Epoch 14/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0131 - bina
ry_accuracy: 0.9935 - val_loss: 0.0923 - val_binary_accuracy: 0.8754
Epoch 15/20
30/30 [==============================] - 0s 17ms/step - loss: 0.0117 - bina
ry_accuracy: 0.9943 - val_loss: 0.0935 - val_binary_accuracy: 0.8735
Epoch 16/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0104 - bina
ry_accuracy: 0.9948 - val_loss: 0.0949 - val_binary_accuracy: 0.8730
Epoch 17/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0094 - bina
ry_accuracy: 0.9956 - val_loss: 0.0957 - val_binary_accuracy: 0.8736
Epoch 18/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0084 - bina
ry_accuracy: 0.9960 - val_loss: 0.0971 - val_binary_accuracy: 0.8713
Epoch 19/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0076 - bina
ry_accuracy: 0.9967 - val_loss: 0.0975 - val_binary_accuracy: 0.8711
Epoch 20/20
30/30 [==============================] - 0s 17ms/step - loss: 0.0070 - bina
ry_accuracy: 0.9969 - val_loss: 0.0985 - val_binary_accuracy: 0.8706
```





In [83]:

```
model = models.Sequential()
model.add(layers.Dense(16, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=2, batch_size=512)
results = model.evaluate(x_test, y_test)

results
```

```
Epoch 1/2
49/49 [==============================] - 2s 11ms/step - loss: 0.1492 - accu
racy: 0.8221
Epoch 2/2
49/49 [==============================] - 1s 11ms/step - loss: 0.0818 - accu
racy: 0.9093
782/782 [==============================] - 2s 2ms/step - loss: 0.0926 - acc
uracy: 0.8840
```

Out[83]:

```
[0.09262750297784805, 0.8840000033378601]
```

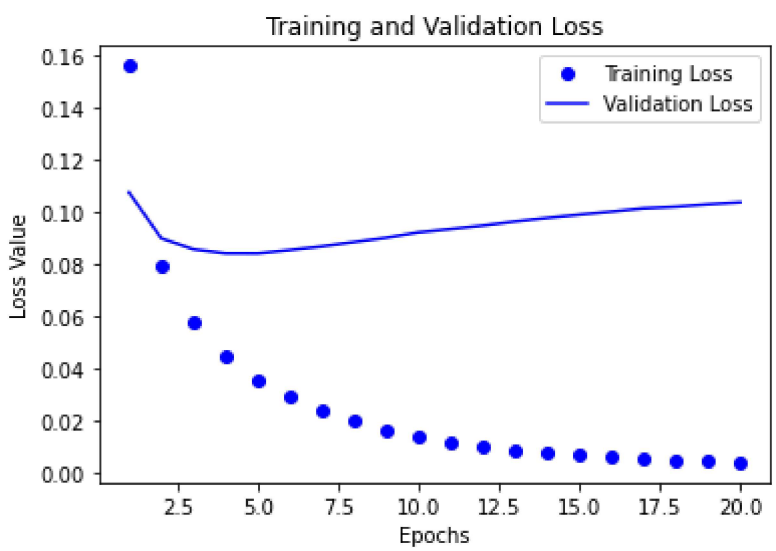
In [84]:

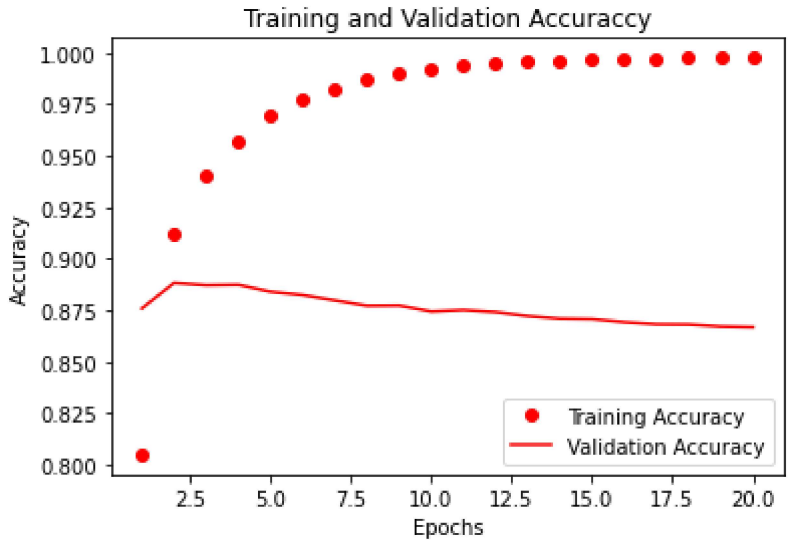
```
#implemented one hideden layer with 32 neurons and mse loss function

from keras import models
from tensorflow.keras import layers

model = models.Sequential()
model.add(layers.Dense(32, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

from keras import optimizers
from keras import losses
from keras import metrics

from tensorflow import keras
from keras import optimizers
from tensorflow.keras import optimizers
from tensorflow.keras import optimizers

model.compile(optimizer='adam',
              loss = losses.mse,
              metrics = [metrics.binary_accuracy])

x_val = x_train[:10000]
partial_x_train = x_train[10000:]

y_val = y_train[:10000]
partial_y_train = y_train[10000:]

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

history_dict = history.history
history_dict.keys()
# Plotting the training and validation loss

import matplotlib.pyplot as plt
%matplotlib inline

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

epochs = range(1, len(loss_values) + 1)

plt.plot(epochs, loss_values, 'bo', label="Training Loss")
plt.plot(epochs, val_loss_values, 'b', label="Validation Loss")

plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss Value')
```

```
plt.legend()

plt.show()


# Plotting the training and validation accuracy
# Training and Validation Accuracy

acc_values = history_dict['binary_accuracy']
val_acc_values = history_dict['val_binary_accuracy']

epochs = range(1, len(loss_values) + 1)

plt.plot(epochs, acc_values, 'ro', label="Training Accuracy")
plt.plot(epochs, val_acc_values, 'r', label="Validation Accuracy")

plt.title('Training and Validation Accuraccy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```

```
Epoch 1/20
30/30 [==============================] - 2s 34ms/step - loss: 0.1561 - b
inary_accuracy: 0.8049 - val_loss: 0.1076 - val_binary_accuracy: 0.8759
Epoch 2/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0792 - b
inary_accuracy: 0.9123 - val_loss: 0.0900 - val_binary_accuracy: 0.8883
Epoch 3/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0576 - b
inary_accuracy: 0.9405 - val_loss: 0.0858 - val_binary_accuracy: 0.8872
Epoch 4/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0449 - b
inary_accuracy: 0.9571 - val_loss: 0.0843 - val_binary_accuracy: 0.8874
Epoch 5/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0359 - b
inary_accuracy: 0.9693 - val_loss: 0.0842 - val_binary_accuracy: 0.8840
Epoch 6/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0292 - b
inary_accuracy: 0.9777 - val_loss: 0.0855 - val_binary_accuracy: 0.8824
Epoch 7/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0241 - b
inary_accuracy: 0.9825 - val_loss: 0.0869 - val_binary_accuracy: 0.8798
Epoch 8/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0200 - b
inary_accuracy: 0.9867 - val_loss: 0.0886 - val_binary_accuracy: 0.8772
Epoch 9/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0168 - b
inary_accuracy: 0.9905 - val_loss: 0.0902 - val_binary_accuracy: 0.8773
Epoch 10/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0142 - b
inary_accuracy: 0.9921 - val_loss: 0.0923 - val_binary_accuracy: 0.8744
Epoch 11/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0120 - b
```

```
inary_accuracy: 0.9936 - val_loss: 0.0936 - val_binary_accuracy: 0.8751
Epoch 12/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0102 - b
inary_accuracy: 0.9949 - val_loss: 0.0949 - val_binary_accuracy: 0.8741
Epoch 13/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0089 - b
inary_accuracy: 0.9956 - val_loss: 0.0965 - val_binary_accuracy: 0.8722
Epoch 14/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0078 - b
inary_accuracy: 0.9962 - val_loss: 0.0978 - val_binary_accuracy: 0.8709
Epoch 15/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0068 - b
inary_accuracy: 0.9967 - val_loss: 0.0991 - val_binary_accuracy: 0.8707
Epoch 16/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0062 - b
inary_accuracy: 0.9972 - val_loss: 0.1003 - val_binary_accuracy: 0.8692
Epoch 17/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0055 - b
inary_accuracy: 0.9973 - val_loss: 0.1015 - val_binary_accuracy: 0.8682
Epoch 18/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0050 - b
inary_accuracy: 0.9975 - val_loss: 0.1021 - val_binary_accuracy: 0.8681
Epoch 19/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0045 - b
inary_accuracy: 0.9975 - val_loss: 0.1031 - val_binary_accuracy: 0.8671
Epoch 20/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0042 - b
inary_accuracy: 0.9977 - val_loss: 0.1038 - val_binary_accuracy: 0.8668
```





In [85]:

```
model = models.Sequential()
model.add(layers.Dense(32, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=2, batch_size=512)
results = model.evaluate(x_test, y_test)

results
```

```
Epoch 1/2
49/49 [==============================] - 1s 12ms/step - loss: 0.1373 - accu
racy: 0.8320
Epoch 2/2
49/49 [==============================] - 1s 12ms/step - loss: 0.0710 - accu
racy: 0.9183
782/782 [==============================] - 2s 3ms/step - loss: 0.0881 - acc
uracy: 0.8873
```

Out[85]:

```
[0.0880509689450264, 0.8873199820518494]
```

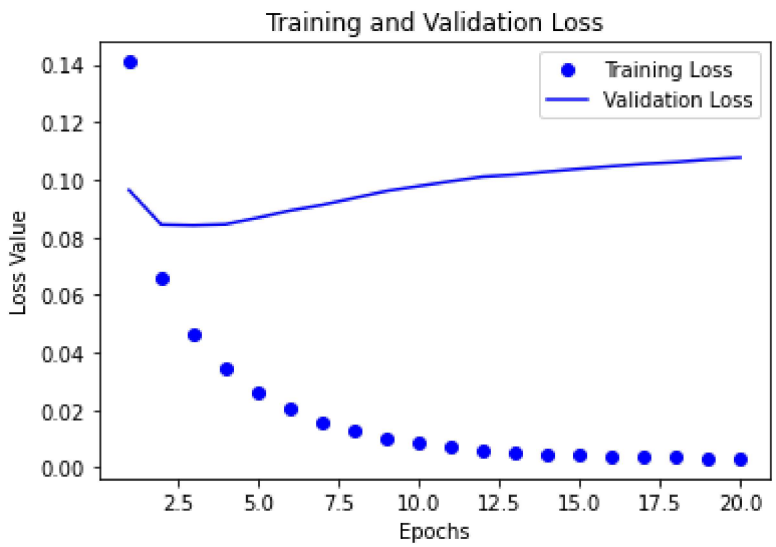
In [86]:

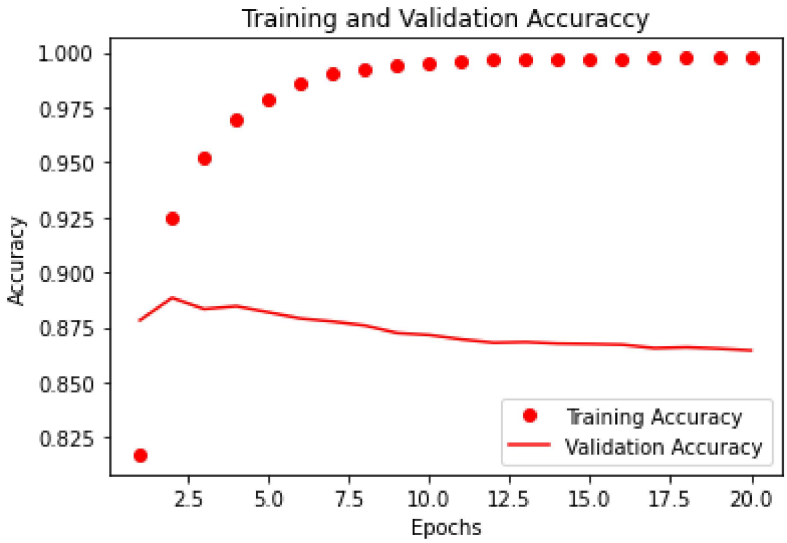
```
#implemented one hideden layer with 64 neurons and mse loss function

from keras import models
from tensorflow.keras import layers

model = models.Sequential()
model.add(layers.Dense(64, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

from keras import optimizers
from keras import losses
from keras import metrics

from tensorflow import keras
from keras import optimizers
from tensorflow.keras import optimizers
from tensorflow.keras import optimizers

model.compile(optimizer='adam',
              loss = losses.mse,
              metrics = [metrics.binary_accuracy])

x_val = x_train[:10000]
partial_x_train = x_train[10000:]

y_val = y_train[:10000]
partial_y_train = y_train[10000:]

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

history_dict = history.history
history_dict.keys()
# Plotting the training and validation loss

import matplotlib.pyplot as plt
%matplotlib inline

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

epochs = range(1, len(loss_values) + 1)

plt.plot(epochs, loss_values, 'bo', label="Training Loss")
plt.plot(epochs, val_loss_values, 'b', label="Validation Loss")

plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss Value')
```

```
plt.legend()

plt.show()


# Plotting the training and validation accuracy
# Training and Validation Accuracy

acc_values = history_dict['binary_accuracy']
val_acc_values = history_dict['val_binary_accuracy']

epochs = range(1, len(loss_values) + 1)

plt.plot(epochs, acc_values, 'ro', label="Training Accuracy")
plt.plot(epochs, val_acc_values, 'r', label="Validation Accuracy")

plt.title('Training and Validation Accuraccy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```

```
Epoch 1/20
30/30 [==============================] - 3s 64ms/step - loss: 0.1410 - bina
ry_accuracy: 0.8171 - val_loss: 0.0964 - val_binary_accuracy: 0.8782
Epoch 2/20
30/30 [==============================] - 1s 38ms/step - loss: 0.0661 - bina
ry_accuracy: 0.9249 - val_loss: 0.0844 - val_binary_accuracy: 0.8884
Epoch 3/20
30/30 [==============================] - 1s 42ms/step - loss: 0.0462 - bina
ry_accuracy: 0.9521 - val_loss: 0.0842 - val_binary_accuracy: 0.8833
Epoch 4/20
30/30 [==============================] - 1s 39ms/step - loss: 0.0346 - bina
ry_accuracy: 0.9691 - val_loss: 0.0845 - val_binary_accuracy: 0.8846
Epoch 5/20
30/30 [==============================] - 1s 36ms/step - loss: 0.0263 - bina
ry_accuracy: 0.9791 - val_loss: 0.0868 - val_binary_accuracy: 0.8818
Epoch 6/20
30/30 [==============================] - 1s 34ms/step - loss: 0.0203 - bina
ry_accuracy: 0.9859 - val_loss: 0.0893 - val_binary_accuracy: 0.8790
Epoch 7/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0160 - bina
ry_accuracy: 0.9902 - val_loss: 0.0912 - val_binary_accuracy: 0.8776
Epoch 8/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0127 - bina
ry_accuracy: 0.9924 - val_loss: 0.0936 - val_binary_accuracy: 0.8758
Epoch 9/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0103 - bina
ry_accuracy: 0.9941 - val_loss: 0.0961 - val_binary_accuracy: 0.8724
Epoch 10/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0084 - bina
ry_accuracy: 0.9953 - val_loss: 0.0978 - val_binary_accuracy: 0.8715
Epoch 11/20
30/30 [==============================] - 1s 32ms/step - loss: 0.0071 - bina
```

```
ry_accuracy: 0.9962 - val_loss: 0.0994 - val_binary_accuracy: 0.8695
Epoch 12/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0061 - bina
ry_accuracy: 0.9967 - val_loss: 0.1010 - val_binary_accuracy: 0.8680
Epoch 13/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0053 - bina
ry_accuracy: 0.9969 - val_loss: 0.1018 - val_binary_accuracy: 0.8683
Epoch 14/20
30/30 [==============================] - 1s 28ms/step - loss: 0.0048 - bina
ry_accuracy: 0.9969 - val_loss: 0.1028 - val_binary_accuracy: 0.8676
Epoch 15/20
30/30 [==============================] - 1s 27ms/step - loss: 0.0044 - bina
ry_accuracy: 0.9971 - val_loss: 0.1038 - val_binary_accuracy: 0.8674
Epoch 16/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0040 - bina
ry_accuracy: 0.9972 - val_loss: 0.1047 - val_binary_accuracy: 0.8671
Epoch 17/20
30/30 [==============================] - 1s 32ms/step - loss: 0.0038 - bina
ry_accuracy: 0.9975 - val_loss: 0.1055 - val_binary_accuracy: 0.8655
Epoch 18/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0035 - bina
ry_accuracy: 0.9975 - val_loss: 0.1061 - val_binary_accuracy: 0.8659
Epoch 19/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0033 - bina
ry_accuracy: 0.9976 - val_loss: 0.1070 - val_binary_accuracy: 0.8653
Epoch 20/20
30/30 [==============================] - 1s 28ms/step - loss: 0.0032 - bina
ry_accuracy: 0.9976 - val_loss: 0.1077 - val_binary_accuracy: 0.8645
```




Training and Validation Accuraccy
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
Accuracy
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epochs
Training Accuracy
Validation Accuracy

In [87]:

```
model = models.Sequential()
model.add(layers.Dense(64, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=4, batch_size=512)
results = model.evaluate(x_test, y_test)

results
```

```
Epoch 1/4
49/49 [==============================] - 2s 22ms/step - loss: 0.1198 - accu
racy: 0.8427
Epoch 2/4
49/49 [==============================] - 1s 20ms/step - loss: 0.0601 - accu
racy: 0.9273
Epoch 3/4
49/49 [==============================] - 1s 19ms/step - loss: 0.0452 - accu
racy: 0.9496
Epoch 4/4
49/49 [==============================] - 1s 19ms/step - loss: 0.0356 - accu
racy: 0.9632
782/782 [==============================] - 3s 3ms/step - loss: 0.0923 - acc
uracy: 0.8743
```

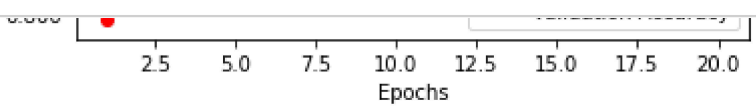
Out[87]:

```
[0.0922648012638092, 0.8742799758911133]
```

In [88]:

```
#implemented one hideden layer with 64 neurons and mse loss function

from keras import models
from tensorflow.keras import layers

model = models.Sequential()
model.add(layers.Dense(64, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

from keras import optimizers
from keras import losses
from keras import metrics

from tensorflow import keras
from keras import optimizers
from tensorflow.keras import optimizers
from tensorflow.keras import optimizers

model.compile(optimizer='adam',
              loss = losses.mse,
              metrics = [metrics.binary_accuracy])

x_val = x_train[:10000]
partial_x_train = x_train[10000:]

y_val = y_train[:10000]
partial_y_train = y_train[10000:]

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

history_dict = history.history
history_dict.keys()
# Plotting the training and validation loss

import matplotlib.pyplot as plt
%matplotlib inline

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

epochs = range(1, len(loss_values) + 1)

plt.plot(epochs, loss_values, 'bo', label="Training Loss")
plt.plot(epochs, val_loss_values, 'b', label="Validation Loss")

plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss Value')
```

```
plt.legend()

plt.show()


# Plotting the training and validation accuracy
# Training and Validation Accuracy

acc_values = history_dict['binary_accuracy']
val_acc_values = history_dict['val_binary_accuracy']

epochs = range(1, len(loss_values) + 1)

plt.plot(epochs, acc_values, 'ro', label="Training Accuracy")
plt.plot(epochs, val_acc_values, 'r', label="Validation Accuracy")

plt.title('Training and Validation Accuraccy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()

model = models.Sequential()
model.add(layers.Dense(64, activation='tanh', input_shape=(10000,)))
layers.Dropout(0.5),
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=4, batch_size=512)
results = model.evaluate(x_test, y_test)

results
```




```
Epoch 1/4
49/49 [==============================] - 2s 19ms/step - loss: 0.1259 - a
ccuracy: 0.8396
Epoch 2/4
49/49 [==============================] - 1s 17ms/step - loss: 0.0623 - a
ccuracy: 0.9252
Epoch 3/4
49/49 [==============================] - 1s 17ms/step - loss: 0.0465 - a
ccuracy: 0.9484
Epoch 4/4
49/49 [==============================] - 1s 17ms/step - loss: 0.0368 - a
ccuracy: 0.9614
782/782 [==============================] - 2s 3ms/step - loss: 0.0911 -
accuracy: 0.8760
```

Out[88]:

```
[0.09106682986021042, 0.8759999871253967]
```

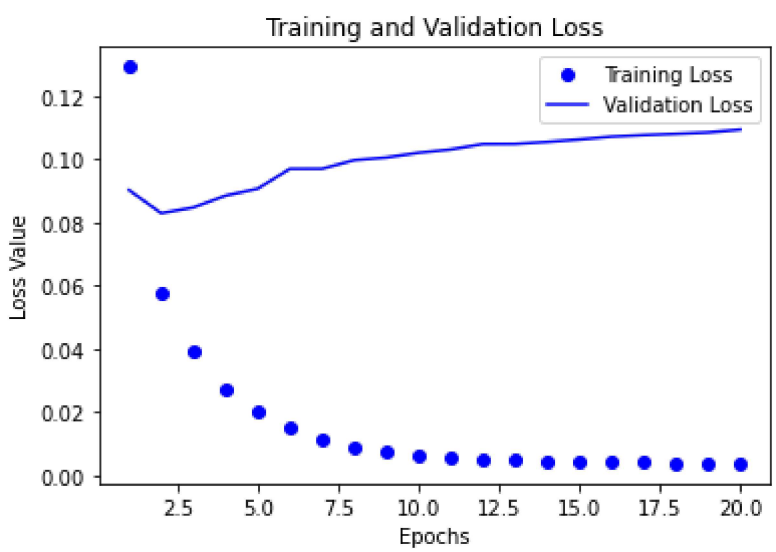
In [89]:

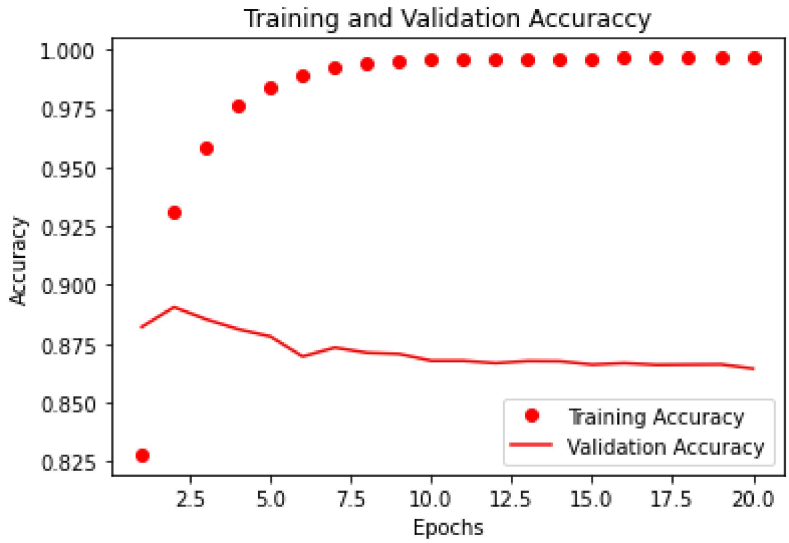
```
#implemented one hideden layer with 128 neurons and mse loss function


from keras import models
from tensorflow.keras import layers

model = models.Sequential()
model.add(layers.Dense(128, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

from keras import optimizers
from keras import losses
from keras import metrics

from tensorflow import keras
from keras import optimizers
from tensorflow.keras import optimizers
from tensorflow.keras import optimizers

model.compile(optimizer='adam',
              loss = losses.mse,
              metrics = [metrics.binary_accuracy])

x_val = x_train[:10000]
partial_x_train = x_train[10000:]

y_val = y_train[:10000]
partial_y_train = y_train[10000:]

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

history_dict = history.history
history_dict.keys()
# Plotting the training and validation loss

import matplotlib.pyplot as plt
%matplotlib inline

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

epochs = range(1, len(loss_values) + 1)

plt.plot(epochs, loss_values, 'bo', label="Training Loss")
plt.plot(epochs, val_loss_values, 'b', label="Validation Loss")

plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
```

```
plt.ylabel('Loss Value')
plt.legend()

plt.show()


# Plotting the training and validation accuracy
# Training and Validation Accuracy

acc_values = history_dict['binary_accuracy']
val_acc_values = history_dict['val_binary_accuracy']

epochs = range(1, len(loss_values) + 1)

plt.plot(epochs, acc_values, 'ro', label="Training Accuracy")
plt.plot(epochs, val_acc_values, 'r', label="Validation Accuracy")

plt.title('Training and Validation Accuraccy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```

```
Epoch 1/20
30/30 [==============================] - 2s 46ms/step - loss: 0.1293 - bina
ry_accuracy: 0.8280 - val_loss: 0.0903 - val_binary_accuracy: 0.8822
Epoch 2/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0578 - bina
ry_accuracy: 0.9309 - val_loss: 0.0830 - val_binary_accuracy: 0.8906
Epoch 3/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0391 - bina
ry_accuracy: 0.9585 - val_loss: 0.0848 - val_binary_accuracy: 0.8854
Epoch 4/20
30/30 [==============================] - 1s 32ms/step - loss: 0.0274 - bina
ry_accuracy: 0.9759 - val_loss: 0.0885 - val_binary_accuracy: 0.8812
Epoch 5/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0204 - bina
ry_accuracy: 0.9838 - val_loss: 0.0907 - val_binary_accuracy: 0.8781
Epoch 6/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0151 - bina
ry_accuracy: 0.9889 - val_loss: 0.0970 - val_binary_accuracy: 0.8696
Epoch 7/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0114 - bina
ry_accuracy: 0.9925 - val_loss: 0.0970 - val_binary_accuracy: 0.8734
Epoch 8/20
30/30 [==============================] - 1s 32ms/step - loss: 0.0088 - bina
ry_accuracy: 0.9943 - val_loss: 0.0997 - val_binary_accuracy: 0.8712
Epoch 9/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0073 - bina
ry_accuracy: 0.9953 - val_loss: 0.1006 - val_binary_accuracy: 0.8707
Epoch 10/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0063 - bina
ry_accuracy: 0.9955 - val_loss: 0.1021 - val_binary_accuracy: 0.8678
```

```
Epoch 11/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0056 - bina
ry_accuracy: 0.9958 - val_loss: 0.1031 - val_binary_accuracy: 0.8678
Epoch 12/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0051 - bina
ry_accuracy: 0.9960 - val_loss: 0.1048 - val_binary_accuracy: 0.8668
Epoch 13/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0048 - bina
ry_accuracy: 0.9961 - val_loss: 0.1048 - val_binary_accuracy: 0.8677
Epoch 14/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0045 - bina
ry_accuracy: 0.9962 - val_loss: 0.1055 - val_binary_accuracy: 0.8676
Epoch 15/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0044 - bina
ry_accuracy: 0.9963 - val_loss: 0.1063 - val_binary_accuracy: 0.8662
Epoch 16/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0042 - bina
ry_accuracy: 0.9963 - val_loss: 0.1072 - val_binary_accuracy: 0.8668
Epoch 17/20
30/30 [==============================] - 1s 32ms/step - loss: 0.0041 - bina
ry_accuracy: 0.9964 - val_loss: 0.1077 - val_binary_accuracy: 0.8661
Epoch 18/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0040 - bina
ry_accuracy: 0.9964 - val_loss: 0.1080 - val_binary_accuracy: 0.8662
Epoch 19/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0039 - bina
ry_accuracy: 0.9965 - val_loss: 0.1085 - val_binary_accuracy: 0.8663
Epoch 20/20
30/30 [==============================] - 1s 32ms/step - loss: 0.0038 - bina
ry_accuracy: 0.9965 - val_loss: 0.1094 - val_binary_accuracy: 0.8645
```





In [90]:

```
model = models.Sequential()
model.add(layers.Dense(128, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=6, batch_size=512)
results = model.evaluate(x_test, y_test)

results
```

```
Epoch 1/2
49/49 [==============================] - 2s 22ms/step - loss: 0.1167 - accu
racy: 0.8446
Epoch 2/2
49/49 [==============================] - 1s 22ms/step - loss: 0.0573 - accu
racy: 0.9294
782/782 [==============================] - 3s 3ms/step - loss: 0.0884 - acc
uracy: 0.8800
```

Out[90]:

```
[0.08843938261270523, 0.8799999952316284]
```

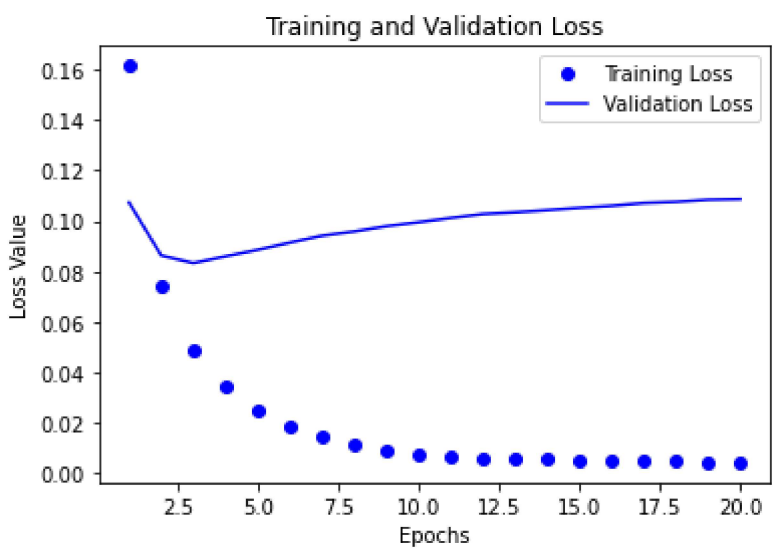
In [91]:

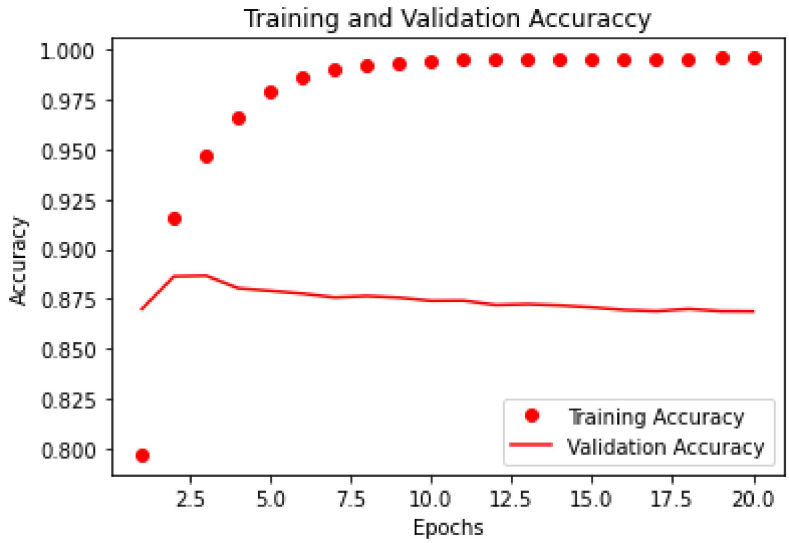
```
#implemented two hidden layer with 16 neurons and mse loss function


from keras import models
from tensorflow.keras import layers

model = models.Sequential()
model.add(layers.Dense(16, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(16, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

from keras import optimizers
from keras import losses
from keras import metrics

from tensorflow import keras
from keras import optimizers
from tensorflow.keras import optimizers
from tensorflow.keras import optimizers

model.compile(optimizer='adam',
              loss = losses.mse,
              metrics = [metrics.binary_accuracy])

x_val = x_train[:10000]
partial_x_train = x_train[10000:]

y_val = y_train[:10000]
partial_y_train = y_train[10000:]

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

history_dict = history.history
history_dict.keys()
# Plotting the training and validation loss

import matplotlib.pyplot as plt
%matplotlib inline

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

epochs = range(1, len(loss_values) + 1)

plt.plot(epochs, loss_values, 'bo', label="Training Loss")
plt.plot(epochs, val_loss_values, 'b', label="Validation Loss")

plt.title('Training and Validation Loss')
```

```
plt.xlabel('Epochs')
plt.ylabel('Loss Value')
plt.legend()

plt.show()


# Plotting the training and validation accuracy
# Training and Validation Accuracy

acc_values = history_dict['binary_accuracy']
val_acc_values = history_dict['val_binary_accuracy']

epochs = range(1, len(loss_values) + 1)

plt.plot(epochs, acc_values, 'ro', label="Training Accuracy")
plt.plot(epochs, val_acc_values, 'r', label="Validation Accuracy")

plt.title('Training and Validation Accuraccy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```

```
Epoch 1/20
30/30 [==============================] - 3s 59ms/step - loss: 0.1616 - b
inary_accuracy: 0.7970 - val_loss: 0.1073 - val_binary_accuracy: 0.8700
Epoch 2/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0744 - b
inary_accuracy: 0.9156 - val_loss: 0.0863 - val_binary_accuracy: 0.8864
Epoch 3/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0488 - b
inary_accuracy: 0.9467 - val_loss: 0.0833 - val_binary_accuracy: 0.8867
Epoch 4/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0346 - b
inary_accuracy: 0.9664 - val_loss: 0.0860 - val_binary_accuracy: 0.8804
Epoch 5/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0250 - b
inary_accuracy: 0.9787 - val_loss: 0.0886 - val_binary_accuracy: 0.8791
Epoch 6/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0187 - b
inary_accuracy: 0.9861 - val_loss: 0.0914 - val_binary_accuracy: 0.8777
Epoch 7/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0142 - b
inary_accuracy: 0.9899 - val_loss: 0.0941 - val_binary_accuracy: 0.8758
Epoch 8/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0110 - b
inary_accuracy: 0.9921 - val_loss: 0.0958 - val_binary_accuracy: 0.8765
Epoch 9/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0089 - b
inary_accuracy: 0.9937 - val_loss: 0.0979 - val_binary_accuracy: 0.8757
Epoch 10/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0075 - b
inary_accuracy: 0.9945 - val_loss: 0.0995 - val_binary_accuracy: 0.8741
```

```
Epoch 11/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0066 - b
inary_accuracy: 0.9949 - val_loss: 0.1012 - val_binary_accuracy: 0.8742
Epoch 12/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0060 - b
inary_accuracy: 0.9951 - val_loss: 0.1027 - val_binary_accuracy: 0.8720
Epoch 13/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0057 - b
inary_accuracy: 0.9952 - val_loss: 0.1034 - val_binary_accuracy: 0.8724
Epoch 14/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0054 - b
inary_accuracy: 0.9953 - val_loss: 0.1043 - val_binary_accuracy: 0.8718
Epoch 15/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0052 - b
inary_accuracy: 0.9954 - val_loss: 0.1052 - val_binary_accuracy: 0.8708
Epoch 16/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0050 - b
inary_accuracy: 0.9955 - val_loss: 0.1060 - val_binary_accuracy: 0.8695
Epoch 17/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0048 - b
inary_accuracy: 0.9956 - val_loss: 0.1071 - val_binary_accuracy: 0.8689
Epoch 18/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0046 - b
inary_accuracy: 0.9957 - val_loss: 0.1076 - val_binary_accuracy: 0.8700
Epoch 19/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0044 - b
inary_accuracy: 0.9959 - val_loss: 0.1084 - val_binary_accuracy: 0.8689
Epoch 20/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0042 - b
inary_accuracy: 0.9961 - val_loss: 0.1087 - val_binary_accuracy: 0.8688
```





In [92]:

```
model = models.Sequential()
model.add(layers.Dense(16, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(16, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=2, batch_size=512)
results = model.evaluate(x_test, y_test)

results
```

```
Epoch 1/2
49/49 [==============================] - 1s 11ms/step - loss: 0.1393 - accu
racy: 0.8234
Epoch 2/2
49/49 [==============================] - 1s 11ms/step - loss: 0.0652 - accu
racy: 0.9204
782/782 [==============================] - 2s 2ms/step - loss: 0.0854 - acc
uracy: 0.8848
```

Out[92]:

```
[0.08544862270355225, 0.8848000168800354]
```

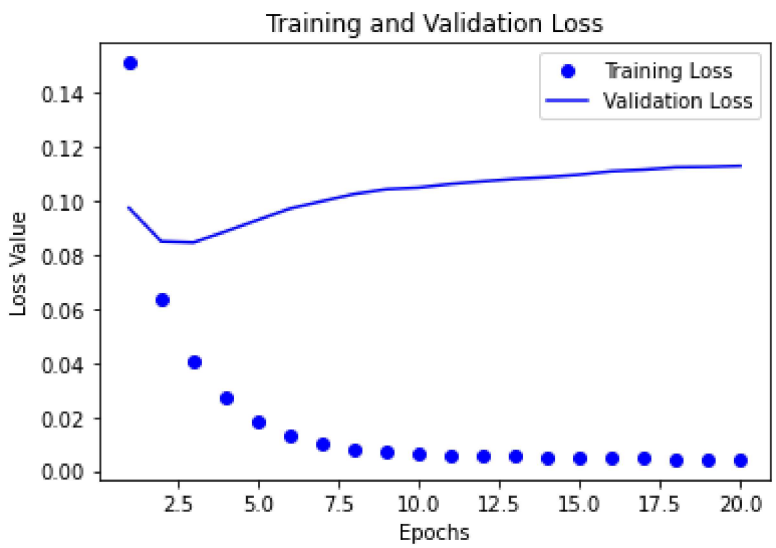
In [93]:

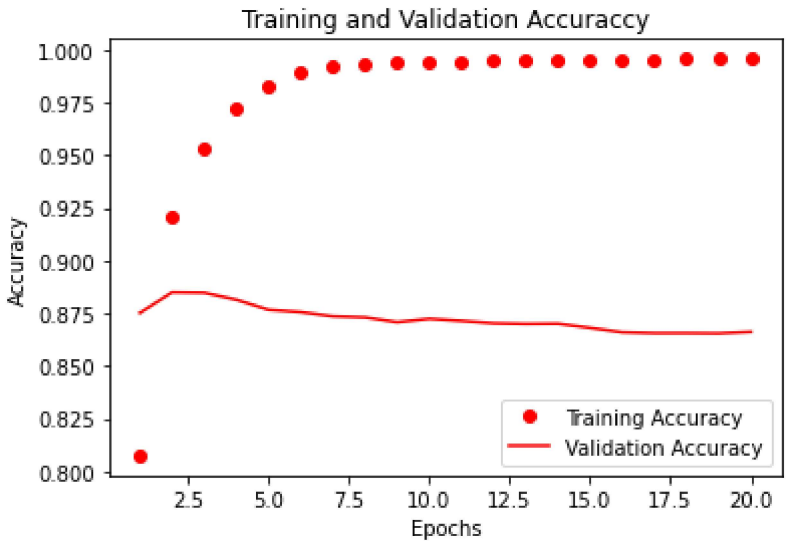
```
#implemented two hideden layer with 16 neurons and mse loss function


from keras import models
from tensorflow.keras import layers

model = models.Sequential()
model.add(layers.Dense(32, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(32, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

from keras import optimizers
from keras import losses
from keras import metrics

from tensorflow import keras
from keras import optimizers
from tensorflow.keras import optimizers
from tensorflow.keras import optimizers

model.compile(optimizer='adam',
              loss = losses.mse,
              metrics = [metrics.binary_accuracy])

x_val = x_train[:10000]
partial_x_train = x_train[10000:]

y_val = y_train[:10000]
partial_y_train = y_train[10000:]

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

history_dict = history.history
history_dict.keys()
# Plotting the training and validation loss

import matplotlib.pyplot as plt
%matplotlib inline

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

epochs = range(1, len(loss_values) + 1)

plt.plot(epochs, loss_values, 'bo', label="Training Loss")
plt.plot(epochs, val_loss_values, 'b', label="Validation Loss")

plt.title('Training and Validation Loss')
```

```
plt.xlabel('Epochs')
plt.ylabel('Loss Value')
plt.legend()

plt.show()


# Plotting the training and validation accuracy
# Training and Validation Accuracy

acc_values = history_dict['binary_accuracy']
val_acc_values = history_dict['val_binary_accuracy']

epochs = range(1, len(loss_values) + 1)

plt.plot(epochs, acc_values, 'ro', label="Training Accuracy")
plt.plot(epochs, val_acc_values, 'r', label="Validation Accuracy")

plt.title('Training and Validation Accuraccy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```

```
Epoch 1/20
30/30 [==============================] - 1s 30ms/step - loss: 0.1511 - bina
ry_accuracy: 0.8075 - val_loss: 0.0973 - val_binary_accuracy: 0.8752
Epoch 2/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0640 - bina
ry_accuracy: 0.9211 - val_loss: 0.0850 - val_binary_accuracy: 0.8850
Epoch 3/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0404 - bina
ry_accuracy: 0.9533 - val_loss: 0.0846 - val_binary_accuracy: 0.8848
Epoch 4/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0272 - bina
ry_accuracy: 0.9724 - val_loss: 0.0887 - val_binary_accuracy: 0.8815
Epoch 5/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0186 - bina
ry_accuracy: 0.9831 - val_loss: 0.0929 - val_binary_accuracy: 0.8767
Epoch 6/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0132 - bina
ry_accuracy: 0.9894 - val_loss: 0.0971 - val_binary_accuracy: 0.8756
Epoch 7/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0101 - bina
ry_accuracy: 0.9920 - val_loss: 0.0999 - val_binary_accuracy: 0.8736
Epoch 8/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0082 - bina
ry_accuracy: 0.9931 - val_loss: 0.1025 - val_binary_accuracy: 0.8731
Epoch 9/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0070 - bina
ry_accuracy: 0.9940 - val_loss: 0.1043 - val_binary_accuracy: 0.8708
Epoch 10/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0062 - bina
ry_accuracy: 0.9945 - val_loss: 0.1049 - val_binary_accuracy: 0.8724
Epoch 11/20
```

```
30/30 [==============================] - 1s 21ms/step - loss: 0.0058 - bina
ry_accuracy: 0.9947 - val_loss: 0.1063 - val_binary_accuracy: 0.8714
Epoch 12/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0056 - bina
ry_accuracy: 0.9947 - val_loss: 0.1072 - val_binary_accuracy: 0.8703
Epoch 13/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0054 - bina
ry_accuracy: 0.9949 - val_loss: 0.1081 - val_binary_accuracy: 0.8700
Epoch 14/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0051 - bina
ry_accuracy: 0.9951 - val_loss: 0.1088 - val_binary_accuracy: 0.8701
Epoch 15/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0050 - bina
ry_accuracy: 0.9952 - val_loss: 0.1097 - val_binary_accuracy: 0.8681
Epoch 16/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0048 - bina
ry_accuracy: 0.9954 - val_loss: 0.1110 - val_binary_accuracy: 0.8660
Epoch 17/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0046 - bina
ry_accuracy: 0.9955 - val_loss: 0.1115 - val_binary_accuracy: 0.8656
Epoch 18/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0044 - bina
ry_accuracy: 0.9958 - val_loss: 0.1124 - val_binary_accuracy: 0.8656
Epoch 19/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0043 - bina
ry_accuracy: 0.9959 - val_loss: 0.1126 - val_binary_accuracy: 0.8655
Epoch 20/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0042 - bina
ry_accuracy: 0.9959 - val_loss: 0.1129 - val_binary_accuracy: 0.8662
```




Training and Validation Accuraccy
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
Accuracy
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epochs
Training Accuracy
Validation Accuracy

In [94]:

```
model = models.Sequential()
model.add(layers.Dense(32, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(32, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=2, batch_size=512)
results = model.evaluate(x_test, y_test)

results
```

```
Epoch 1/2
49/49 [==============================] - 1s 15ms/step - loss: 0.1265 - accu
racy: 0.8337
Epoch 2/2
49/49 [==============================] - 1s 13ms/step - loss: 0.0583 - accu
racy: 0.9260
782/782 [==============================] - 2s 3ms/step - loss: 0.0864 - acc
uracy: 0.8822
```

Out[94]:

```
[0.0864405557513237, 0.8822000026702881]
```

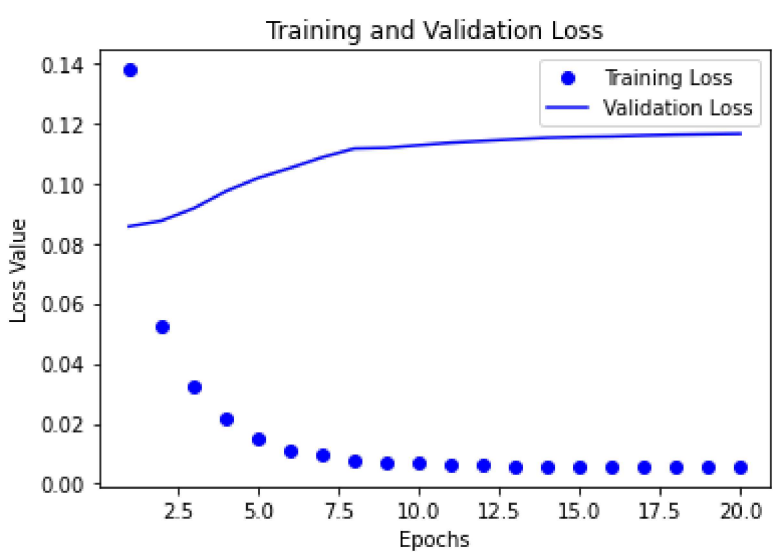
In [95]:

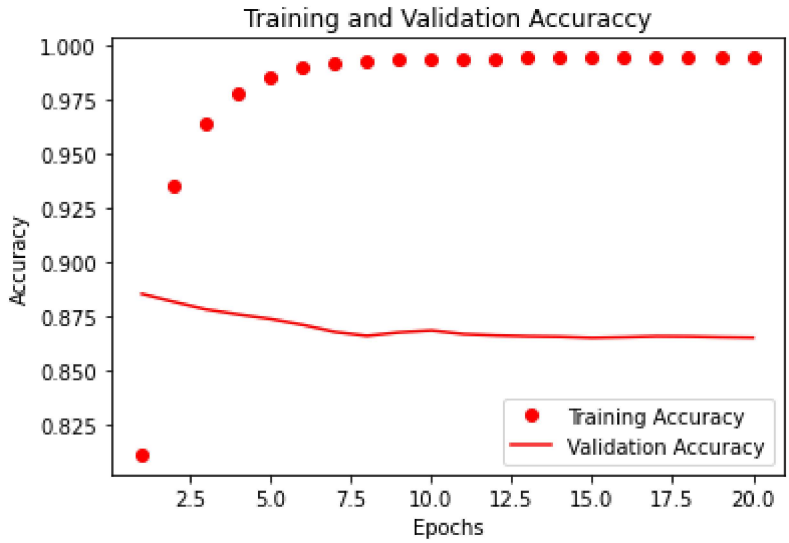
```
#implemented two hideden layer with 64 neurons and mse loss function


from keras import models
from tensorflow.keras import layers

model = models.Sequential()
model.add(layers.Dense(64, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(64, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

from keras import optimizers
from keras import losses
from keras import metrics

from tensorflow import keras
from keras import optimizers
from tensorflow.keras import optimizers
from tensorflow.keras import optimizers

model.compile(optimizer='adam',
              loss = losses.mse,
              metrics = [metrics.binary_accuracy])

x_val = x_train[:10000]
partial_x_train = x_train[10000:]

y_val = y_train[:10000]
partial_y_train = y_train[10000:]

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

history_dict = history.history
history_dict.keys()
# Plotting the training and validation loss

import matplotlib.pyplot as plt
%matplotlib inline

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

epochs = range(1, len(loss_values) + 1)

plt.plot(epochs, loss_values, 'bo', label="Training Loss")
plt.plot(epochs, val_loss_values, 'b', label="Validation Loss")

plt.title('Training and Validation Loss')
```

```
plt.xlabel('Epochs')
plt.ylabel('Loss Value')
plt.legend()

plt.show()


# Plotting the training and validation accuracy
# Training and Validation Accuracy

acc_values = history_dict['binary_accuracy']
val_acc_values = history_dict['val_binary_accuracy']

epochs = range(1, len(loss_values) + 1)

plt.plot(epochs, acc_values, 'ro', label="Training Accuracy")
plt.plot(epochs, val_acc_values, 'r', label="Validation Accuracy")

plt.title('Training and Validation Accuraccy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```

```
Epoch 1/20
30/30 [==============================] - 2s 59ms/step - loss: 0.1377 - bina
ry_accuracy: 0.8111 - val_loss: 0.0856 - val_binary_accuracy: 0.8853
Epoch 2/20
30/30 [==============================] - 1s 34ms/step - loss: 0.0522 - bina
ry_accuracy: 0.9351 - val_loss: 0.0875 - val_binary_accuracy: 0.8816
Epoch 3/20
30/30 [==============================] - 1s 34ms/step - loss: 0.0324 - bina
ry_accuracy: 0.9641 - val_loss: 0.0916 - val_binary_accuracy: 0.8781
Epoch 4/20
30/30 [==============================] - 1s 36ms/step - loss: 0.0214 - bina
ry_accuracy: 0.9783 - val_loss: 0.0972 - val_binary_accuracy: 0.8758
Epoch 5/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0150 - bina
ry_accuracy: 0.9857 - val_loss: 0.1017 - val_binary_accuracy: 0.8737
Epoch 6/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0112 - bina
ry_accuracy: 0.9899 - val_loss: 0.1050 - val_binary_accuracy: 0.8710
Epoch 7/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0094 - bina
ry_accuracy: 0.9917 - val_loss: 0.1086 - val_binary_accuracy: 0.8677
Epoch 8/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0079 - bina
ry_accuracy: 0.9929 - val_loss: 0.1115 - val_binary_accuracy: 0.8659
Epoch 9/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0071 - bina
ry_accuracy: 0.9933 - val_loss: 0.1117 - val_binary_accuracy: 0.8675
Epoch 10/20
30/30 [==============================] - 1s 29ms/step - loss: 0.0067 - bina
ry_accuracy: 0.9935 - val_loss: 0.1126 - val_binary_accuracy: 0.8684
Epoch 11/20
```

```
30/30 [==============================] - 1s 29ms/step - loss: 0.0063 - bina
ry_accuracy: 0.9939 - val_loss: 0.1134 - val_binary_accuracy: 0.8667
Epoch 12/20
30/30 [==============================] - 1s 29ms/step - loss: 0.0062 - bina
ry_accuracy: 0.9939 - val_loss: 0.1140 - val_binary_accuracy: 0.8661
Epoch 13/20
30/30 [==============================] - 1s 28ms/step - loss: 0.0059 - bina
ry_accuracy: 0.9941 - val_loss: 0.1146 - val_binary_accuracy: 0.8657
Epoch 14/20
30/30 [==============================] - 1s 28ms/step - loss: 0.0058 - bina
ry_accuracy: 0.9943 - val_loss: 0.1151 - val_binary_accuracy: 0.8655
Epoch 15/20
30/30 [==============================] - 1s 28ms/step - loss: 0.0058 - bina
ry_accuracy: 0.9943 - val_loss: 0.1154 - val_binary_accuracy: 0.8650
Epoch 16/20
30/30 [==============================] - 1s 28ms/step - loss: 0.0058 - bina
ry_accuracy: 0.9943 - val_loss: 0.1155 - val_binary_accuracy: 0.8653
Epoch 17/20
30/30 [==============================] - 1s 28ms/step - loss: 0.0057 - bina
ry_accuracy: 0.9943 - val_loss: 0.1159 - val_binary_accuracy: 0.8657
Epoch 18/20
30/30 [==============================] - 1s 28ms/step - loss: 0.0057 - bina
ry_accuracy: 0.9943 - val_loss: 0.1161 - val_binary_accuracy: 0.8656
Epoch 19/20
30/30 [==============================] - 1s 28ms/step - loss: 0.0057 - bina
ry_accuracy: 0.9943 - val_loss: 0.1163 - val_binary_accuracy: 0.8653
Epoch 20/20
30/30 [==============================] - 1s 29ms/step - loss: 0.0057 - bina
ry_accuracy: 0.9943 - val_loss: 0.1164 - val_binary_accuracy: 0.8651
```





In [96]:

```
model = models.Sequential()
model.add(layers.Dense(64, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(64, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=2, batch_size=512)
results = model.evaluate(x_test, y_test)

results
```

```
Epoch 1/2
49/49 [==============================] - 2s 18ms/step - loss: 0.1170 - accu
racy: 0.8397
Epoch 2/2
49/49 [==============================] - 1s 18ms/step - loss: 0.0537 - accu
racy: 0.9315
782/782 [==============================] - 2s 3ms/step - loss: 0.0928 - acc
uracy: 0.8760
```

Out[96]:

```
[0.09281282126903534, 0.8759999871253967]
```

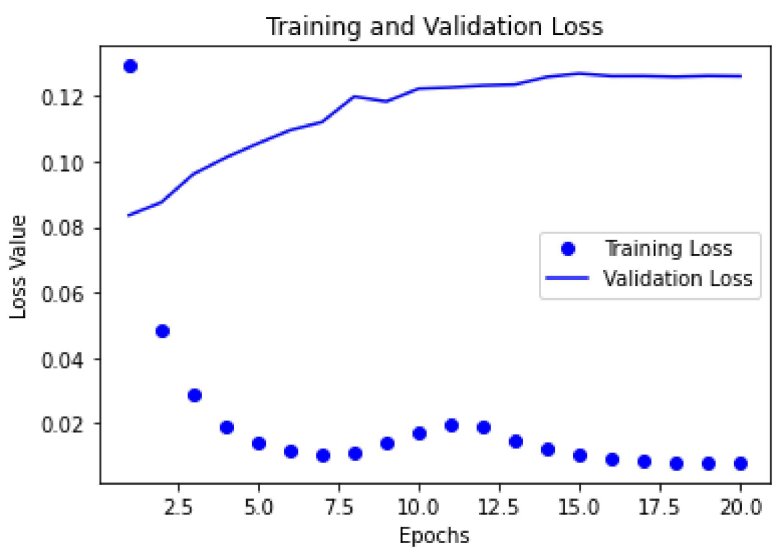
In [97]:

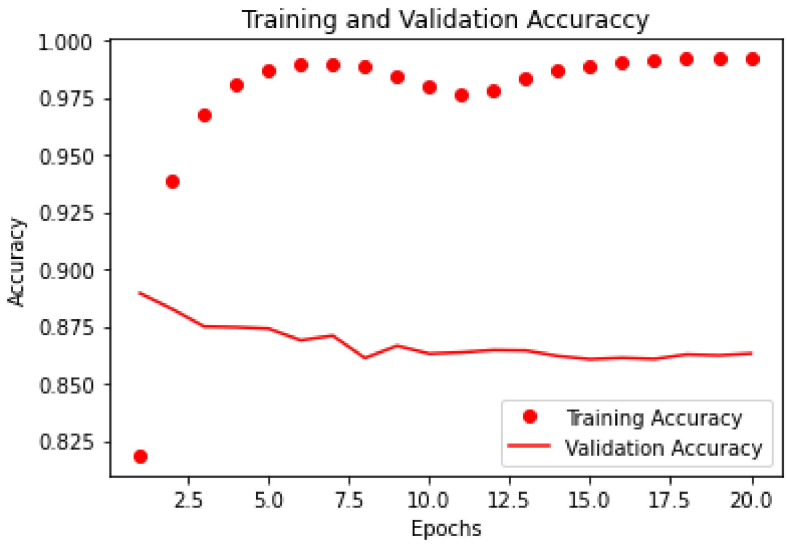
```
#implemented two hideden layer with 128 neurons and mse loss function


from keras import models
from tensorflow.keras import layers

model = models.Sequential()
model.add(layers.Dense(128, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(128, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

from keras import optimizers
from keras import losses
from keras import metrics

from tensorflow import keras
from keras import optimizers
from tensorflow.keras import optimizers
from tensorflow.keras import optimizers

model.compile(optimizer='adam',
              loss = losses.mse,
              metrics = [metrics.binary_accuracy])

x_val = x_train[:10000]
partial_x_train = x_train[10000:]

y_val = y_train[:10000]
partial_y_train = y_train[10000:]

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

history_dict = history.history
history_dict.keys()
# Plotting the training and validation loss

import matplotlib.pyplot as plt
%matplotlib inline

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

epochs = range(1, len(loss_values) + 1)

plt.plot(epochs, loss_values, 'bo', label="Training Loss")
plt.plot(epochs, val_loss_values, 'b', label="Validation Loss")

plt.title('Training and Validation Loss')
```

```
plt.xlabel('Epochs')
plt.ylabel('Loss Value')
plt.legend()

plt.show()


# Plotting the training and validation accuracy
# Training and Validation Accuracy

acc_values = history_dict['binary_accuracy']
val_acc_values = history_dict['val_binary_accuracy']

epochs = range(1, len(loss_values) + 1)

plt.plot(epochs, acc_values, 'ro', label="Training Accuracy")
plt.plot(epochs, val_acc_values, 'r', label="Validation Accuracy")

plt.title('Training and Validation Accuraccy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```

```
Epoch 1/20
30/30 [==============================] - 2s 42ms/step - loss: 0.1293 - b
inary_accuracy: 0.8185 - val_loss: 0.0836 - val_binary_accuracy: 0.8895
Epoch 2/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0484 - b
inary_accuracy: 0.9382 - val_loss: 0.0875 - val_binary_accuracy: 0.8826
Epoch 3/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0291 - b
inary_accuracy: 0.9677 - val_loss: 0.0962 - val_binary_accuracy: 0.8749
Epoch 4/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0188 - b
inary_accuracy: 0.9805 - val_loss: 0.1012 - val_binary_accuracy: 0.8746
Epoch 5/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0139 - b
inary_accuracy: 0.9867 - val_loss: 0.1055 - val_binary_accuracy: 0.8740
Epoch 6/20
30/30 [==============================] - 1s 34ms/step - loss: 0.0115 - b
inary_accuracy: 0.9893 - val_loss: 0.1096 - val_binary_accuracy: 0.8689
Epoch 7/20
30/30 [==============================] - 1s 34ms/step - loss: 0.0108 - b
inary_accuracy: 0.9895 - val_loss: 0.1121 - val_binary_accuracy: 0.8709
Epoch 8/20
30/30 [==============================] - 1s 34ms/step - loss: 0.0112 - b
inary_accuracy: 0.9887 - val_loss: 0.1199 - val_binary_accuracy: 0.8611
Epoch 9/20
30/30 [==============================] - 1s 34ms/step - loss: 0.0143 - b
inary_accuracy: 0.9844 - val_loss: 0.1184 - val_binary_accuracy: 0.8665
Epoch 10/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0173 - b
inary_accuracy: 0.9801 - val_loss: 0.1223 - val_binary_accuracy: 0.8630
```

```
Epoch 11/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0199 - b
inary_accuracy: 0.9761 - val_loss: 0.1227 - val_binary_accuracy: 0.8636
Epoch 12/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0188 - b
inary_accuracy: 0.9781 - val_loss: 0.1233 - val_binary_accuracy: 0.8646
Epoch 13/20
30/30 [==============================] - 1s 34ms/step - loss: 0.0146 - b
inary_accuracy: 0.9836 - val_loss: 0.1236 - val_binary_accuracy: 0.8644
Epoch 14/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0126 - b
inary_accuracy: 0.9865 - val_loss: 0.1259 - val_binary_accuracy: 0.8620
Epoch 15/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0104 - b
inary_accuracy: 0.9890 - val_loss: 0.1269 - val_binary_accuracy: 0.8606
Epoch 16/20
30/30 [==============================] - 1s 37ms/step - loss: 0.0091 - b
inary_accuracy: 0.9906 - val_loss: 0.1262 - val_binary_accuracy: 0.8612
Epoch 17/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0085 - b
inary_accuracy: 0.9916 - val_loss: 0.1262 - val_binary_accuracy: 0.8607
Epoch 18/20
30/30 [==============================] - 1s 32ms/step - loss: 0.0082 - b
inary_accuracy: 0.9919 - val_loss: 0.1259 - val_binary_accuracy: 0.8627
Epoch 19/20
30/30 [==============================] - 1s 32ms/step - loss: 0.0081 - b
inary_accuracy: 0.9919 - val_loss: 0.1262 - val_binary_accuracy: 0.8623
Epoch 20/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0081 - b
inary_accuracy: 0.9919 - val_loss: 0.1261 - val_binary_accuracy: 0.8631
```




Training and Validation Accuraccy
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
Accuracy
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epochs
Training Accuracy
Validation Accuracy

In [98]:

```
model = models.Sequential()
model.add(layers.Dense(128, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(128, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=2, batch_size=512)
results = model.evaluate(x_test, y_test)

results
```

```
Epoch 1/2
49/49 [==============================] - 2s 23ms/step - loss: 0.1085 - accu
racy: 0.8509
Epoch 2/2
49/49 [==============================] - 1s 22ms/step - loss: 0.0525 - accu
racy: 0.9333
782/782 [==============================] - 3s 3ms/step - loss: 0.0932 - acc
uracy: 0.8767
```

Out[98]:

```
[0.09319932758808136, 0.8766800165176392]
```

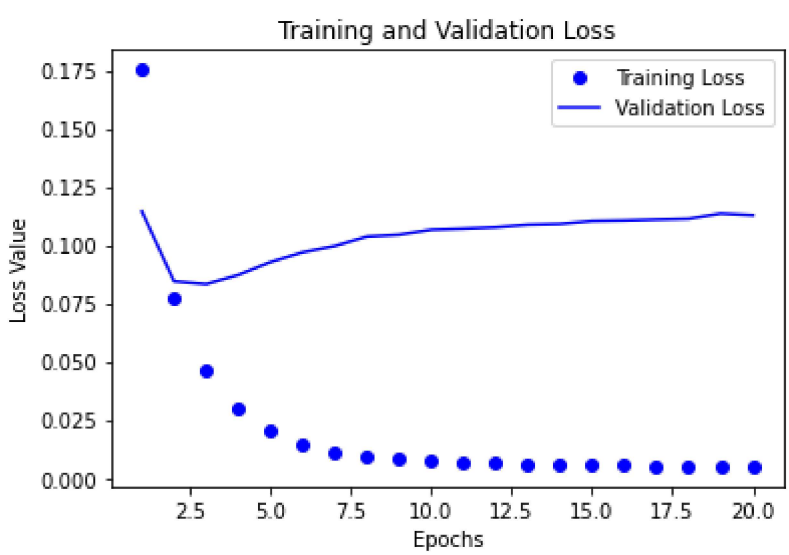
In [99]:

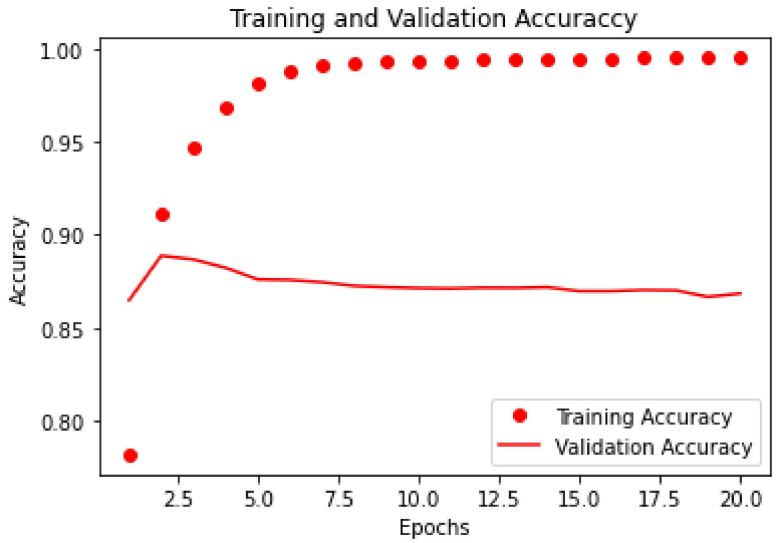
```
#implemented three hideden layer with 16 neurons and mse loss function


from keras import models
from tensorflow.keras import layers

model = models.Sequential()
model.add(layers.Dense(16, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(16, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(16, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

from keras import optimizers
from keras import losses
from keras import metrics

from tensorflow import keras
from keras import optimizers
from tensorflow.keras import optimizers
from tensorflow.keras import optimizers

model.compile(optimizer='adam',
              loss = losses.mse,
              metrics = [metrics.binary_accuracy])

x_val = x_train[:10000]
partial_x_train = x_train[10000:]

y_val = y_train[:10000]
partial_y_train = y_train[10000:]

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

history_dict = history.history
history_dict.keys()
# Plotting the training and validation loss

import matplotlib.pyplot as plt
%matplotlib inline

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

epochs = range(1, len(loss_values) + 1)

plt.plot(epochs, loss_values, 'bo', label="Training Loss")
plt.plot(epochs, val_loss_values, 'b', label="Validation Loss")
```

```
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss Value')
plt.legend()

plt.show()


# Plotting the training and validation accuracy
# Training and Validation Accuracy

acc_values = history_dict['binary_accuracy']
val_acc_values = history_dict['val_binary_accuracy']

epochs = range(1, len(loss_values) + 1)

plt.plot(epochs, acc_values, 'ro', label="Training Accuracy")
plt.plot(epochs, val_acc_values, 'r', label="Validation Accuracy")

plt.title('Training and Validation Accuraccy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```

```
Epoch 1/20
30/30 [==============================] - 2s 35ms/step - loss: 0.1753 - bina
ry_accuracy: 0.7819 - val_loss: 0.1144 - val_binary_accuracy: 0.8648
Epoch 2/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0774 - bina
ry_accuracy: 0.9115 - val_loss: 0.0844 - val_binary_accuracy: 0.8887
Epoch 3/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0464 - bina
ry_accuracy: 0.9473 - val_loss: 0.0832 - val_binary_accuracy: 0.8867
Epoch 4/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0302 - bina
ry_accuracy: 0.9687 - val_loss: 0.0872 - val_binary_accuracy: 0.8822
Epoch 5/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0203 - bina
ry_accuracy: 0.9818 - val_loss: 0.0927 - val_binary_accuracy: 0.8760
Epoch 6/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0146 - bina
ry_accuracy: 0.9873 - val_loss: 0.0969 - val_binary_accuracy: 0.8757
Epoch 7/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0110 - bina
ry_accuracy: 0.9906 - val_loss: 0.0996 - val_binary_accuracy: 0.8745
Epoch 8/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0090 - bina
ry_accuracy: 0.9921 - val_loss: 0.1037 - val_binary_accuracy: 0.8725
Epoch 9/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0080 - bina
ry_accuracy: 0.9928 - val_loss: 0.1044 - val_binary_accuracy: 0.8718
Epoch 10/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0072 - bina
ry_accuracy: 0.9934 - val_loss: 0.1066 - val_binary_accuracy: 0.8713
Epoch 11/20
```

```
30/30 [==============================] - 1s 18ms/step - loss: 0.0067 - bina
ry_accuracy: 0.9937 - val_loss: 0.1070 - val_binary_accuracy: 0.8711
Epoch 12/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0063 - bina
ry_accuracy: 0.9940 - val_loss: 0.1076 - val_binary_accuracy: 0.8715
Epoch 13/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0061 - bina
ry_accuracy: 0.9941 - val_loss: 0.1087 - val_binary_accuracy: 0.8714
Epoch 14/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0058 - bina
ry_accuracy: 0.9944 - val_loss: 0.1091 - val_binary_accuracy: 0.8718
Epoch 15/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0056 - bina
ry_accuracy: 0.9946 - val_loss: 0.1103 - val_binary_accuracy: 0.8697
Epoch 16/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0054 - bina
ry_accuracy: 0.9947 - val_loss: 0.1105 - val_binary_accuracy: 0.8697
Epoch 17/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0053 - bina
ry_accuracy: 0.9949 - val_loss: 0.1109 - val_binary_accuracy: 0.8703
Epoch 18/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0051 - bina
ry_accuracy: 0.9950 - val_loss: 0.1113 - val_binary_accuracy: 0.8701
Epoch 19/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0051 - bina
ry_accuracy: 0.9950 - val_loss: 0.1135 - val_binary_accuracy: 0.8666
Epoch 20/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0049 - bina
ry_accuracy: 0.9951 - val_loss: 0.1128 - val_binary_accuracy: 0.8684
```





In [100]:

```
model = models.Sequential()
model.add(layers.Dense(16, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(16, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(16, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=2, batch_size=512)
results = model.evaluate(x_test, y_test)

results
```

```
Epoch 1/2
49/49 [==============================] - 1s 12ms/step - loss: 0.1347 - accu
racy: 0.8326
Epoch 2/2
49/49 [==============================] - 1s 11ms/step - loss: 0.0610 - accu
racy: 0.9238
782/782 [==============================] - 2s 2ms/step - loss: 0.0868 - acc
uracy: 0.8830
```

Out[100]:

```
[0.08675289899110794, 0.8829600214958191]
```

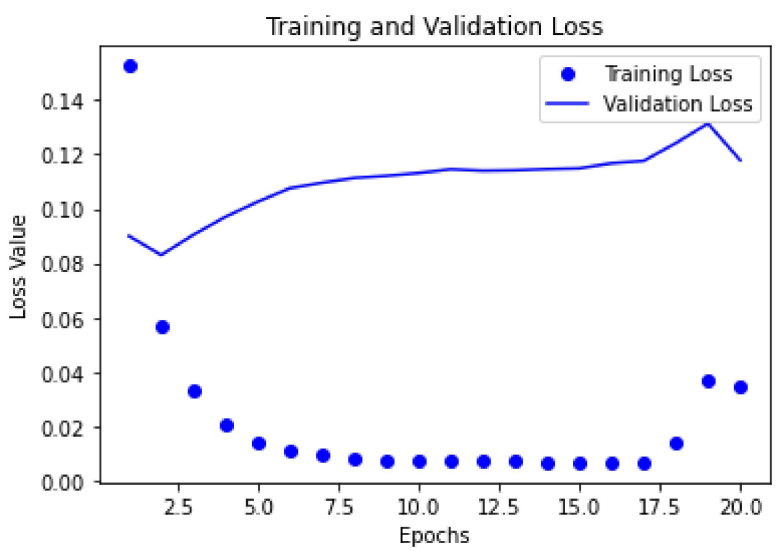
In [101]:

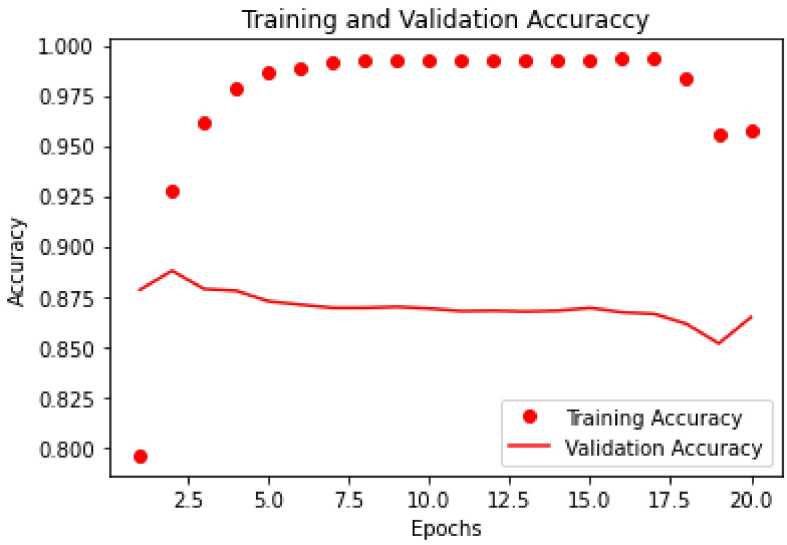
```
#implemented three hideden layer with 32 neurons and mse loss function


from keras import models
from tensorflow.keras import layers

model = models.Sequential()
model.add(layers.Dense(32, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(32, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(32, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

from keras import optimizers
from keras import losses
from keras import metrics

from tensorflow import keras
from keras import optimizers
from tensorflow.keras import optimizers
from tensorflow.keras import optimizers

model.compile(optimizer='adam',
              loss = losses.mse,
              metrics = [metrics.binary_accuracy])

x_val = x_train[:10000]
partial_x_train = x_train[10000:]

y_val = y_train[:10000]
partial_y_train = y_train[10000:]

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

history_dict = history.history
history_dict.keys()
# Plotting the training and validation loss

import matplotlib.pyplot as plt
%matplotlib inline

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

epochs = range(1, len(loss_values) + 1)

plt.plot(epochs, loss_values, 'bo', label="Training Loss")
plt.plot(epochs, val_loss_values, 'b', label="Validation Loss")
```

```
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss Value')
plt.legend()

plt.show()


# Plotting the training and validation accuracy
# Training and Validation Accuracy

acc_values = history_dict['binary_accuracy']
val_acc_values = history_dict['val_binary_accuracy']

epochs = range(1, len(loss_values) + 1)

plt.plot(epochs, acc_values, 'ro', label="Training Accuracy")
plt.plot(epochs, val_acc_values, 'r', label="Validation Accuracy")

plt.title('Training and Validation Accuraccy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```

```
Epoch 1/20
30/30 [==============================] - 1s 30ms/step - loss: 0.1526 - b
inary_accuracy: 0.7958 - val_loss: 0.0900 - val_binary_accuracy: 0.8783
Epoch 2/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0567 - b
inary_accuracy: 0.9279 - val_loss: 0.0830 - val_binary_accuracy: 0.8878
Epoch 3/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0329 - b
inary_accuracy: 0.9613 - val_loss: 0.0904 - val_binary_accuracy: 0.8787
Epoch 4/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0210 - b
inary_accuracy: 0.9790 - val_loss: 0.0971 - val_binary_accuracy: 0.8778
Epoch 5/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0143 - b
inary_accuracy: 0.9867 - val_loss: 0.1026 - val_binary_accuracy: 0.8726
Epoch 6/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0114 - b
inary_accuracy: 0.9891 - val_loss: 0.1076 - val_binary_accuracy: 0.8709
Epoch 7/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0094 - b
inary_accuracy: 0.9915 - val_loss: 0.1096 - val_binary_accuracy: 0.8693
Epoch 8/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0082 - b
inary_accuracy: 0.9922 - val_loss: 0.1113 - val_binary_accuracy: 0.8693
Epoch 9/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0077 - b
inary_accuracy: 0.9925 - val_loss: 0.1121 - val_binary_accuracy: 0.8698
Epoch 10/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0075 - b
inary_accuracy: 0.9927 - val_loss: 0.1132 - val_binary_accuracy: 0.8690
```

```
Epoch 11/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0072 - b
inary_accuracy: 0.9929 - val_loss: 0.1145 - val_binary_accuracy: 0.8677
Epoch 12/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0071 - b
inary_accuracy: 0.9929 - val_loss: 0.1140 - val_binary_accuracy: 0.8679
Epoch 13/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0071 - b
inary_accuracy: 0.9929 - val_loss: 0.1142 - val_binary_accuracy: 0.8675
Epoch 14/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0070 - b
inary_accuracy: 0.9931 - val_loss: 0.1146 - val_binary_accuracy: 0.8679
Epoch 15/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0069 - b
inary_accuracy: 0.9931 - val_loss: 0.1149 - val_binary_accuracy: 0.8693
Epoch 16/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0068 - b
inary_accuracy: 0.9933 - val_loss: 0.1168 - val_binary_accuracy: 0.8671
Epoch 17/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0067 - b
inary_accuracy: 0.9933 - val_loss: 0.1176 - val_binary_accuracy: 0.8663
Epoch 18/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0138 - b
inary_accuracy: 0.9841 - val_loss: 0.1242 - val_binary_accuracy: 0.8613
Epoch 19/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0369 - b
inary_accuracy: 0.9557 - val_loss: 0.1314 - val_binary_accuracy: 0.8516
Epoch 20/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0347 - b
inary_accuracy: 0.9577 - val_loss: 0.1179 - val_binary_accuracy: 0.8646
```




Training and Validation Accuraccy
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
Accuracy
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epochs
Training Accuracy
Validation Accuracy

In [102]:

```
model = models.Sequential()
model.add(layers.Dense(32, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(32, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(32, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=2, batch_size=512)
results = model.evaluate(x_test, y_test)

results
```

```
Epoch 1/2
49/49 [==============================] - 2s 13ms/step - loss: 0.1197 - accu
racy: 0.8426
Epoch 2/2
49/49 [==============================] - 1s 13ms/step - loss: 0.0551 - accu
racy: 0.9293
782/782 [==============================] - 2s 2ms/step - loss: 0.0918 - acc
uracy: 0.8772
```

Out[102]:

```
[0.09177339822053909, 0.8771600127220154]
```

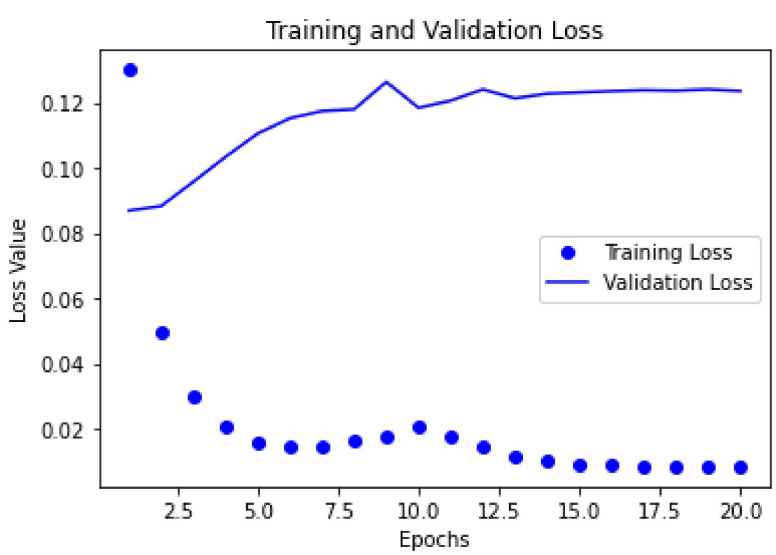
In [103]:

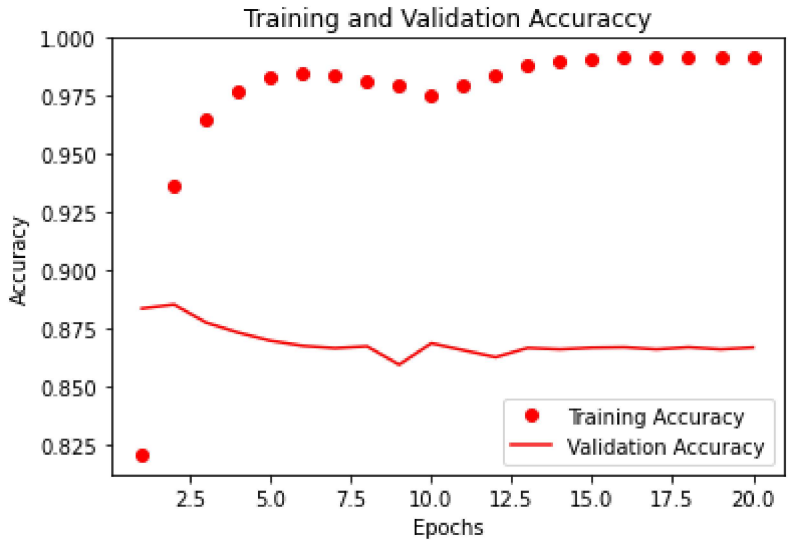
```
#implemented three hideden layer with 64 neurons and mse loss function


from keras import models
from tensorflow.keras import layers

model = models.Sequential()
model.add(layers.Dense(64, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(64, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(64, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

from keras import optimizers
from keras import losses
from keras import metrics

from tensorflow import keras
from keras import optimizers
from tensorflow.keras import optimizers
from tensorflow.keras import optimizers

model.compile(optimizer='adam',
              loss = losses.mse,
              metrics = [metrics.binary_accuracy])

x_val = x_train[:10000]
partial_x_train = x_train[10000:]

y_val = y_train[:10000]
partial_y_train = y_train[10000:]

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

history_dict = history.history
history_dict.keys()
# Plotting the training and validation loss

import matplotlib.pyplot as plt
%matplotlib inline

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

epochs = range(1, len(loss_values) + 1)

plt.plot(epochs, loss_values, 'bo', label="Training Loss")
plt.plot(epochs, val_loss_values, 'b', label="Validation Loss")
```

```
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss Value')
plt.legend()

plt.show()


# Plotting the training and validation accuracy
# Training and Validation Accuracy

acc_values = history_dict['binary_accuracy']
val_acc_values = history_dict['val_binary_accuracy']

epochs = range(1, len(loss_values) + 1)

plt.plot(epochs, acc_values, 'ro', label="Training Accuracy")
plt.plot(epochs, val_acc_values, 'r', label="Validation Accuracy")

plt.title('Training and Validation Accuraccy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```

```
Epoch 1/20
30/30 [==============================] - 2s 43ms/step - loss: 0.1301 - bina
ry_accuracy: 0.8207 - val_loss: 0.0869 - val_binary_accuracy: 0.8836
Epoch 2/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0497 - bina
ry_accuracy: 0.9362 - val_loss: 0.0883 - val_binary_accuracy: 0.8853
Epoch 3/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0297 - bina
ry_accuracy: 0.9651 - val_loss: 0.0957 - val_binary_accuracy: 0.8775
Epoch 4/20
30/30 [==============================] - 1s 32ms/step - loss: 0.0205 - bina
ry_accuracy: 0.9772 - val_loss: 0.1034 - val_binary_accuracy: 0.8732
Epoch 5/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0160 - bina
ry_accuracy: 0.9833 - val_loss: 0.1105 - val_binary_accuracy: 0.8697
Epoch 6/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0147 - bina
ry_accuracy: 0.9844 - val_loss: 0.1152 - val_binary_accuracy: 0.8675
Epoch 7/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0148 - bina
ry_accuracy: 0.9837 - val_loss: 0.1174 - val_binary_accuracy: 0.8665
Epoch 8/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0164 - bina
ry_accuracy: 0.9818 - val_loss: 0.1179 - val_binary_accuracy: 0.8673
Epoch 9/20
30/30 [==============================] - 1s 32ms/step - loss: 0.0174 - bina
ry_accuracy: 0.9795 - val_loss: 0.1263 - val_binary_accuracy: 0.8594
Epoch 10/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0204 - bina
ry_accuracy: 0.9755 - val_loss: 0.1184 - val_binary_accuracy: 0.8686
Epoch 11/20
```

```
30/30 [==============================] - 1s 33ms/step - loss: 0.0177 - bina
ry_accuracy: 0.9795 - val_loss: 0.1206 - val_binary_accuracy: 0.8656
Epoch 12/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0145 - bina
ry_accuracy: 0.9843 - val_loss: 0.1240 - val_binary_accuracy: 0.8626
Epoch 13/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0113 - bina
ry_accuracy: 0.9885 - val_loss: 0.1213 - val_binary_accuracy: 0.8666
Epoch 14/20
30/30 [==============================] - 1s 29ms/step - loss: 0.0101 - bina
ry_accuracy: 0.9897 - val_loss: 0.1227 - val_binary_accuracy: 0.8660
Epoch 15/20
30/30 [==============================] - 1s 28ms/step - loss: 0.0090 - bina
ry_accuracy: 0.9909 - val_loss: 0.1231 - val_binary_accuracy: 0.8667
Epoch 16/20
30/30 [==============================] - 1s 28ms/step - loss: 0.0087 - bina
ry_accuracy: 0.9914 - val_loss: 0.1235 - val_binary_accuracy: 0.8669
Epoch 17/20
30/30 [==============================] - 1s 27ms/step - loss: 0.0087 - bina
ry_accuracy: 0.9913 - val_loss: 0.1238 - val_binary_accuracy: 0.8660
Epoch 18/20
30/30 [==============================] - 1s 28ms/step - loss: 0.0086 - bina
ry_accuracy: 0.9914 - val_loss: 0.1236 - val_binary_accuracy: 0.8669
Epoch 19/20
30/30 [==============================] - 1s 28ms/step - loss: 0.0085 - bina
ry_accuracy: 0.9915 - val_loss: 0.1240 - val_binary_accuracy: 0.8660
Epoch 20/20
30/30 [==============================] - 1s 27ms/step - loss: 0.0085 - bina
ry_accuracy: 0.9915 - val_loss: 0.1235 - val_binary_accuracy: 0.8668
```





In [104]:

```
model = models.Sequential()
model.add(layers.Dense(64, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(64, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(64, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=2, batch_size=512)
results = model.evaluate(x_test, y_test)

results
```

```
Epoch 1/2
49/49 [==============================] - 2s 18ms/step - loss: 0.1114 - accu
racy: 0.8484
Epoch 2/2
49/49 [==============================] - 1s 18ms/step - loss: 0.0516 - accu
racy: 0.9346
782/782 [==============================] - 2s 3ms/step - loss: 0.0945 - acc
uracy: 0.8762
```

Out[104]:

```
[0.09446453303098679, 0.8762000203132629]
```

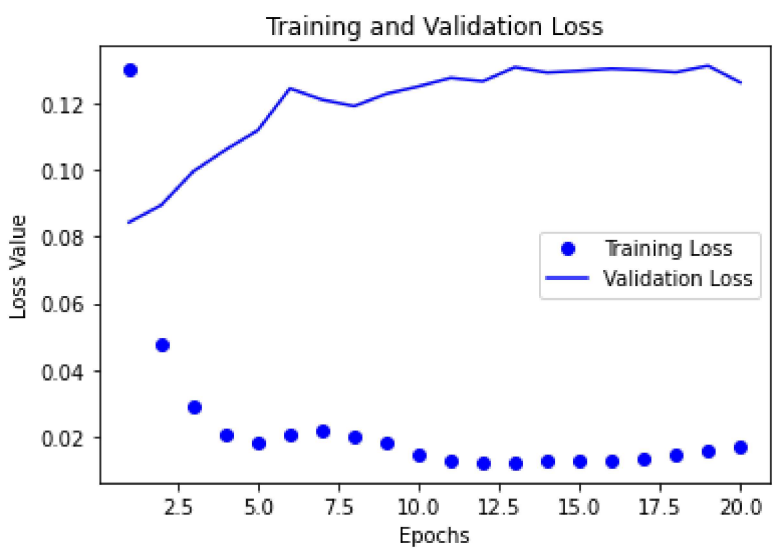
In [105]:

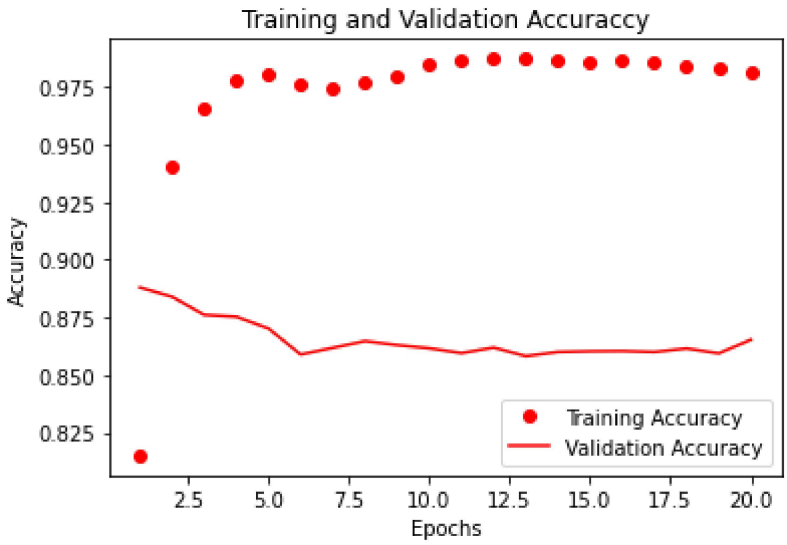
```
#implemented three hideden layer with 128 neurons and mse loss function


from keras import models
from tensorflow.keras import layers

model = models.Sequential()
model.add(layers.Dense(128, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(128, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(128, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

from keras import optimizers
from keras import losses
from keras import metrics

from tensorflow import keras
from keras import optimizers
from tensorflow.keras import optimizers
from tensorflow.keras import optimizers

model.compile(optimizer='adam',
              loss = losses.mse,
              metrics = [metrics.binary_accuracy])

x_val = x_train[:10000]
partial_x_train = x_train[10000:]

y_val = y_train[:10000]
partial_y_train = y_train[10000:]

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

history_dict = history.history
history_dict.keys()
# Plotting training and validation loss

import matplotlib.pyplot as plt
%matplotlib inline

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

epochs = range(1, len(loss_values) + 1)

plt.plot(epochs, loss_values, 'bo', label="Training Loss")
plt.plot(epochs, val_loss_values, 'b', label="Validation Loss")
```

```
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss Value')
plt.legend()

plt.show()


# Plotting the training and validation accuracy
# Training and Validation Accuracy

acc_values = history_dict['binary_accuracy']
val_acc_values = history_dict['val_binary_accuracy']

epochs = range(1, len(loss_values) + 1)

plt.plot(epochs, acc_values, 'ro', label="Training Accuracy")
plt.plot(epochs, val_acc_values, 'r', label="Validation Accuracy")

plt.title('Training and Validation Accuraccy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```

```
Epoch 1/20
30/30 [==============================] - 2s 44ms/step - loss: 0.1303 - b
inary_accuracy: 0.8151 - val_loss: 0.0843 - val_binary_accuracy: 0.8880
Epoch 2/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0478 - b
inary_accuracy: 0.9405 - val_loss: 0.0894 - val_binary_accuracy: 0.8840
Epoch 3/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0293 - b
inary_accuracy: 0.9661 - val_loss: 0.0996 - val_binary_accuracy: 0.8761
Epoch 4/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0205 - b
inary_accuracy: 0.9783 - val_loss: 0.1061 - val_binary_accuracy: 0.8753
Epoch 5/20
30/30 [==============================] - 1s 36ms/step - loss: 0.0180 - b
inary_accuracy: 0.9808 - val_loss: 0.1119 - val_binary_accuracy: 0.8702
Epoch 6/20
30/30 [==============================] - 1s 36ms/step - loss: 0.0207 - b
inary_accuracy: 0.9763 - val_loss: 0.1245 - val_binary_accuracy: 0.8589
Epoch 7/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0221 - b
inary_accuracy: 0.9743 - val_loss: 0.1211 - val_binary_accuracy: 0.8618
Epoch 8/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0202 - b
inary_accuracy: 0.9771 - val_loss: 0.1192 - val_binary_accuracy: 0.8647
Epoch 9/20
30/30 [==============================] - 1s 37ms/step - loss: 0.0183 - b
inary_accuracy: 0.9798 - val_loss: 0.1228 - val_binary_accuracy: 0.8630
Epoch 10/20
30/30 [==============================] - 1s 37ms/step - loss: 0.0146 - b
inary_accuracy: 0.9851 - val_loss: 0.1251 - val_binary_accuracy: 0.8616
```

```
Epoch 11/20
30/30 [==============================] - 1s 34ms/step - loss: 0.0128 - b
inary_accuracy: 0.9868 - val_loss: 0.1276 - val_binary_accuracy: 0.8595
Epoch 12/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0122 - b
inary_accuracy: 0.9873 - val_loss: 0.1266 - val_binary_accuracy: 0.8619
Epoch 13/20
30/30 [==============================] - 1s 39ms/step - loss: 0.0124 - b
inary_accuracy: 0.9872 - val_loss: 0.1308 - val_binary_accuracy: 0.8582
Epoch 14/20
30/30 [==============================] - 1s 40ms/step - loss: 0.0125 - b
inary_accuracy: 0.9870 - val_loss: 0.1292 - val_binary_accuracy: 0.8600
Epoch 15/20
30/30 [==============================] - 1s 39ms/step - loss: 0.0130 - b
inary_accuracy: 0.9861 - val_loss: 0.1298 - val_binary_accuracy: 0.8603
Epoch 16/20
30/30 [==============================] - 1s 37ms/step - loss: 0.0125 - b
inary_accuracy: 0.9869 - val_loss: 0.1304 - val_binary_accuracy: 0.8604
Epoch 17/20
30/30 [==============================] - 1s 34ms/step - loss: 0.0132 - b
inary_accuracy: 0.9860 - val_loss: 0.1300 - val_binary_accuracy: 0.8600
Epoch 18/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0146 - b
inary_accuracy: 0.9841 - val_loss: 0.1293 - val_binary_accuracy: 0.8615
Epoch 19/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0155 - b
inary_accuracy: 0.9829 - val_loss: 0.1313 - val_binary_accuracy: 0.8594
Epoch 20/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0167 - b
inary_accuracy: 0.9811 - val_loss: 0.1263 - val_binary_accuracy: 0.8653
```




Training and Validation Accuraccy
Accuracy
0.975
0.950
0.925
0.900
0.875
0.850
0.825
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epochs
Training Accuracy
Validation Accuracy

In [106]:

```
model = models.Sequential()
model.add(layers.Dense(128, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(128, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(128, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=2, batch_size=512)
results = model.evaluate(x_test, y_test)

results
```

```
Epoch 1/2
49/49 [==============================] - 2s 24ms/step - loss: 0.1069 - accu
racy: 0.8519
Epoch 2/2
49/49 [==============================] - 1s 23ms/step - loss: 0.0512 - accu
racy: 0.9360
782/782 [==============================] - 3s 3ms/step - loss: 0.0974 - acc
uracy: 0.8718
```

Out[106]:

```
[0.09738101810216904, 0.8717600107192993]
```

In [ ]: