AML-Assignment2

March 5, 2023

[1]:
```
import tensorflow
```

[2]:
```
from tensorflow.keras.datasets import imdb
(train_data, train_labels), (test_data, test_labels) = imdb.load_data(
    num_words=10000)
```

[3]:
```
train_data[0]
```

[3]:
```
[1,
 14,
 22,
 16,
 43,
 530,
 973,
 1622,
 1385,
 65,
 458,
 4468,
 66,
 3941,
 4,
 173,
 36,
 256,
 5,
 25,
 100,
 43,
 838,
 112,
 50,
 670,
 2,
 9,
 35,
 480,
 284,
 5,
 150,
 4,
 172,
 112,
 167,
 2,
 336,
 385,
 39,
 4,
 172,
 4536,
 1111,
 17,
 546,
 38,
 13,
 447,
 4,
 192,
 50,
 16,
 6,
 147,
 2025,
 19,
 14,
 22,
 4,
 1920,
 4613,
 469,
 4,
 22,
 71,
 87,
 12,
 16,
 43,
 530,
 38,
 76,
 15,
 13,
 1247,
 4,
 22,
 17,
 515,
 17,
 12,
 16,
 626,
 18,
 2,
 5,
 62,
 386,
 12,
 8,
 316,
 8,
 106,
 5,
 4,
 2223,
 5244,
 16,
 480,
 66,
 3785,
 33,
 4,
 130,
 12,
 16,
 38,
 619,
 5,
 25,
 124,
 51,
 36,
 135,
 48,
 25,
 1415,
 33,
 6,
 22,
 12,
 215,
 28,
 77,
 52,
 5,
 14,
 407,
 16,
 82,
 2,
 8,
 4,
 107,
 117,
 5952,
 15,
 256,
 4,
 2,
 7,
 3766,
 5,
 723,
 36,
 71,
 43,
 530,
 476,
 26,
 400,
 317,
 46,
 7,
 4,
 2,
 1029,
 13,
 104,
 88,
 4,
 381,
 15,
 297,
 98,
 32,
 2071,
 56,
 26,
 141,
 6,
 194,
 7486,
 18,
 4,
 226,
 22,
 21,
 134,
 476,
 26,
 480,
 5,
 144,
 30,
 5535,
 18,
 51,
 36,
 28,
 224,
 92,
 25,
 104,
 4,
 226,
 65,
 16,
 38,
 1334,
 88,
 12,
 16,
 283,
 5,
 16,
 4472,
 113,
 103,
 32,
 15,
 16,
 5345,
 19,
 178,
 32]
```

[4]:
```
train_labels[0]
```

[4]:
```
1
```

[5]:
```
max([max(sequence) for sequence in train_data])
```

[5]:
```
9999
```

[6]:
```
#Decoding reviews back to text
```

[7]:
```
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]])
```

[8]:
```
#Preparinf data set
```

[9]:
```
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)
```

[10]:
```
x_train[0]
```

[10]:
```
array([0., 1., 1., …, 0., 0., 0.])
```

[11]:
```
y_train = np.asarray(train_labels).astype("float32")
y_test = np.asarray(test_labels).astype("float32")
```

[12]:
```
#Buliding Model
```

[13]:
```
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")
])
```

[14]:
```
model.compile(optimizer="adam",
              loss="binary_crossentropy",
                metrics=["accuracy"])
```

[15]:
```
#Validating your approach
```

[16]:
```
x_val = x_train[:10000]
partial_x_train = x_train[10000:]
y_val = y_train[:10000]
partial_y_train = y_train[10000:]
```

[17]:
```
# Training Model
```

[18]:
```
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 26ms/step - loss: 0.5555 - accuracy:
0.7772 - val_loss: 0.4066 - val_accuracy: 0.8587
Epoch 2/20
30/30 [==============================] - 0s 13ms/step - loss: 0.3057 - accuracy:
0.8963 - val_loss: 0.2988 - val_accuracy: 0.8826
Epoch 3/20
30/30 [==============================] - 0s 16ms/step - loss: 0.2071 - accuracy:
0.9288 - val_loss: 0.2763 - val_accuracy: 0.8905
Epoch 4/20
30/30 [==============================] - 0s 13ms/step - loss: 0.1540 - accuracy:
0.9506 - val_loss: 0.2815 - val_accuracy: 0.8878
Epoch 5/20
30/30 [==============================] - 0s 14ms/step - loss: 0.1180 - accuracy:
0.9654 - val_loss: 0.2937 - val_accuracy: 0.8850
Epoch 6/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0914 - accuracy:
0.9753 - val_loss: 0.3150 - val_accuracy: 0.8818
Epoch 7/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0709 - accuracy:
0.9846 - val_loss: 0.3397 - val_accuracy: 0.8805
Epoch 8/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0554 - accuracy:
0.9894 - val_loss: 0.3659 - val_accuracy: 0.8799
Epoch 9/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0427 - accuracy:
0.9933 - val_loss: 0.3953 - val_accuracy: 0.8785
Epoch 10/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0331 - accuracy:
0.9961 - val_loss: 0.4252 - val_accuracy: 0.8752
Epoch 11/20
```

```
30/30 [==============================] - 0s 12ms/step - loss: 0.0255 - accuracy:
0.9975 - val_loss: 0.4531 - val_accuracy: 0.8744
Epoch 12/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0195 - accuracy:
0.9990 - val_loss: 0.4810 - val_accuracy: 0.8731
Epoch 13/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0156 - accuracy:
0.9994 - val_loss: 0.5174 - val_accuracy: 0.8672
Epoch 14/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0125 - accuracy:
0.9997 - val_loss: 0.5347 - val_accuracy: 0.8715
Epoch 15/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0101 - accuracy:
0.9999 - val_loss: 0.5584 - val_accuracy: 0.8698
Epoch 16/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0083 - accuracy:
0.9999 - val_loss: 0.5810 - val_accuracy: 0.8689
Epoch 17/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0069 - accuracy:
0.9999 - val_loss: 0.6024 - val_accuracy: 0.8678
Epoch 18/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0058 - accuracy:
0.9999 - val_loss: 0.6231 - val_accuracy: 0.8674
Epoch 19/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0049 - accuracy:
0.9999 - val_loss: 0.6399 - val_accuracy: 0.8683
Epoch 20/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0041 - accuracy:
0.9999 - val_loss: 0.6593 - val_accuracy: 0.8677
```

[19]:

```
# Compiling the model
```

[20]:

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

[20]:

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

[21]:

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

[22]:

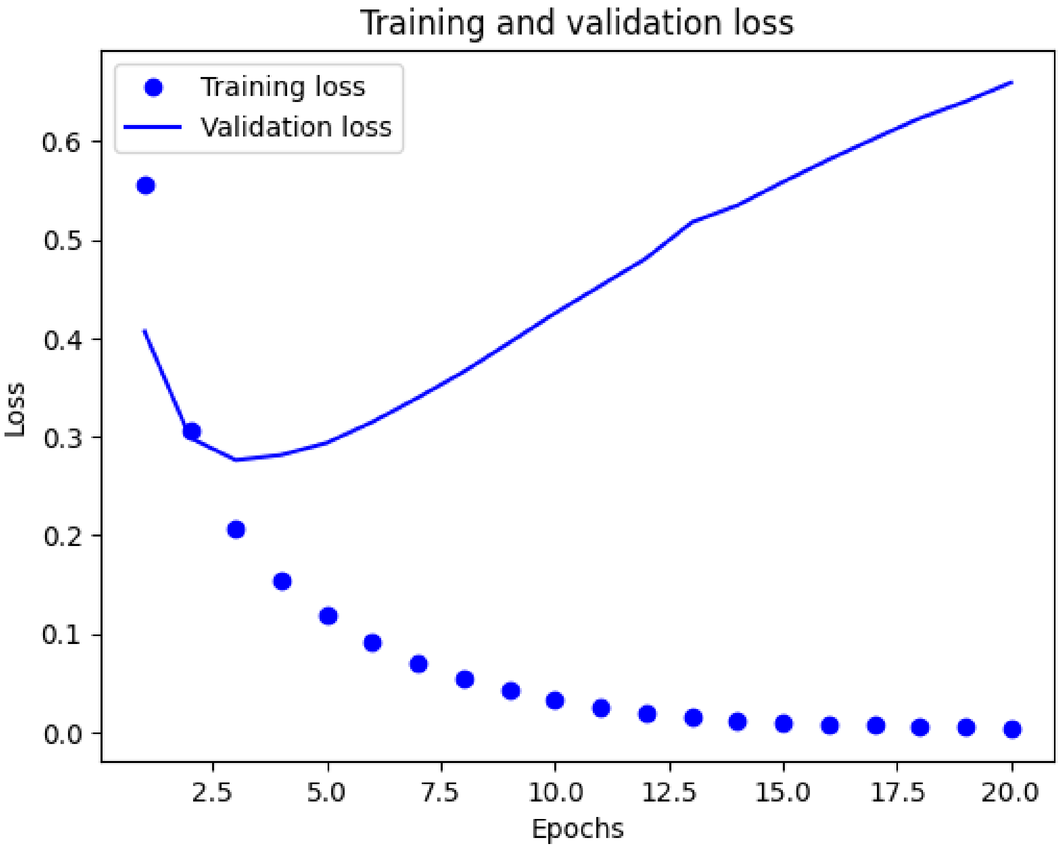
```
pip install matplotlib
```

```
Requirement already satisfied: matplotlib in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (3.7.1)
Requirement already satisfied: cycler>=0.10 in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: packaging>=20.0 in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from matplotlib) (22.0)
Requirement already satisfied: python-dateutil>=2.7 in
```

```
c:\users\akhil\anaconda3\new folder\envs\tensorflow\lib\site-packages (from
matplotlib) (2.8.2)
Requirement already satisfied: importlib-resources>=3.2.0 in
c:\users\akhil\anaconda3\new folder\envs\tensorflow\lib\site-packages (from
matplotlib) (5.12.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from matplotlib) (1.0.7)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from matplotlib) (4.38.0)
Requirement already satisfied: numpy>=1.20 in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from matplotlib) (1.23.5)
Requirement already satisfied: pillow>=6.2.0 in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: zipp>=3.1.0 in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from importlib-
resources>=3.2.0->matplotlib) (3.11.0)
Requirement already satisfied: six>=1.5 in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from python-dateutil>=2.7->matplotlib)
(1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

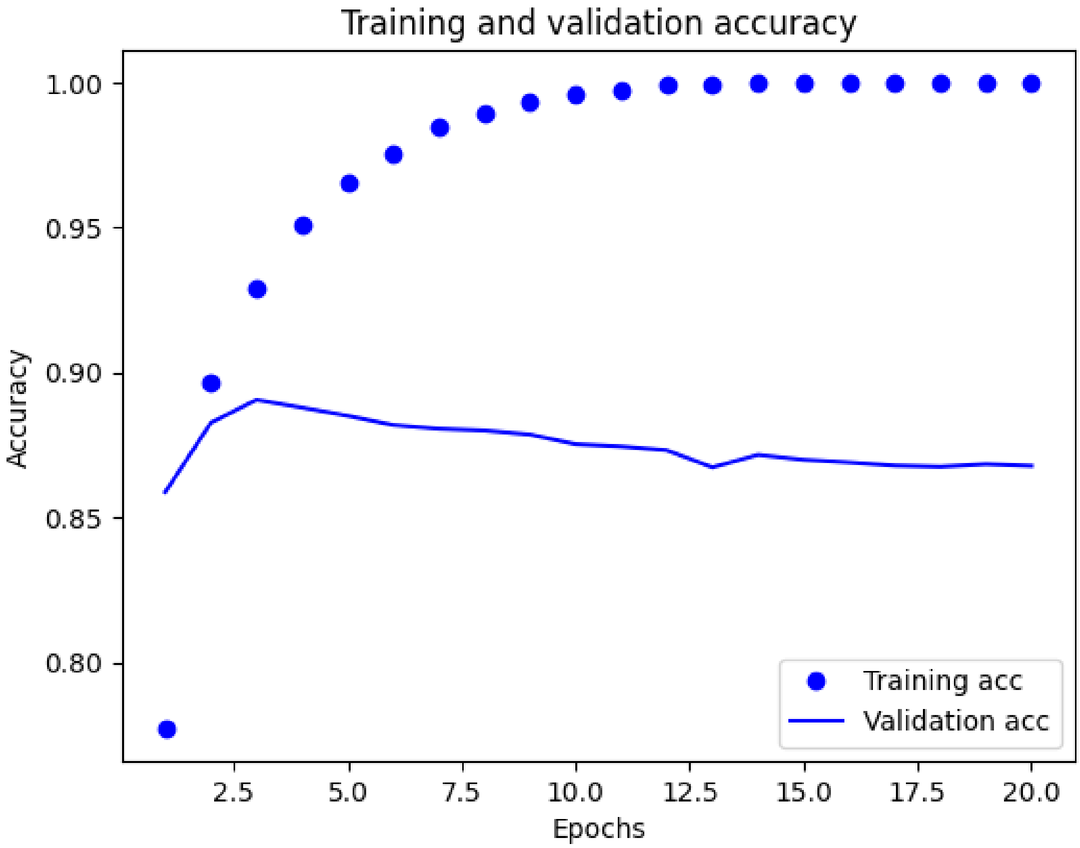
[23]:

```python
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()
```



[24]:
```
# Plotting the training and accuracy
```

[25]:
```
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()
```



[26]:

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

[27]:

```
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 6ms/step - loss: 0.4594 - accuracy:
0.8199
Epoch 2/4
49/49 [==============================] - 0s 6ms/step - loss: 0.2595 - accuracy:
0.9099
Epoch 3/4
```

```
49/49 [==============================] - 0s 7ms/step - loss: 0.2004 - accuracy:
0.9289
Epoch 4/4
49/49 [==============================] - 0s 7ms/step - loss: 0.1683 - accuracy:
0.9389
782/782 [==============================] - 24s 30ms/step - loss: 0.2933 -
accuracy: 0.8828
```

[28]:
```
results
```

[28]:
```
[0.29334139823913574, 0.8828399777412415]
```

[29]:
```
#Using a trained model to generate predictions on new data
```

[30]:
```
model.predict(x_test)
```

```
782/782 [==============================] - 22s 28ms/step
```

[30]:
```
array([[0.18415162],
       [0.9997871 ],
       [0.85542995],
       …,
       [0.09114674],
       [0.05132174],
       [0.46824807]], dtype=float32)
```

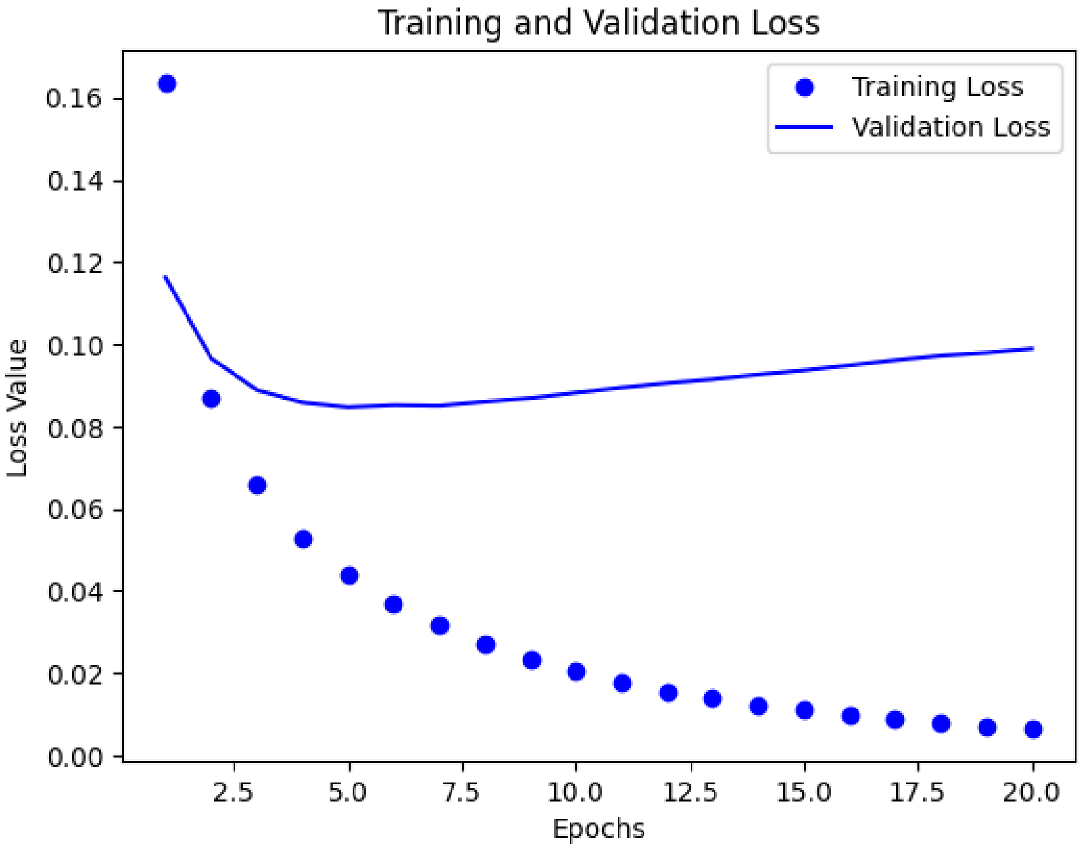
[31]:
```
#implemented one hideden layer with 16 neurons and mse loss function

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
```

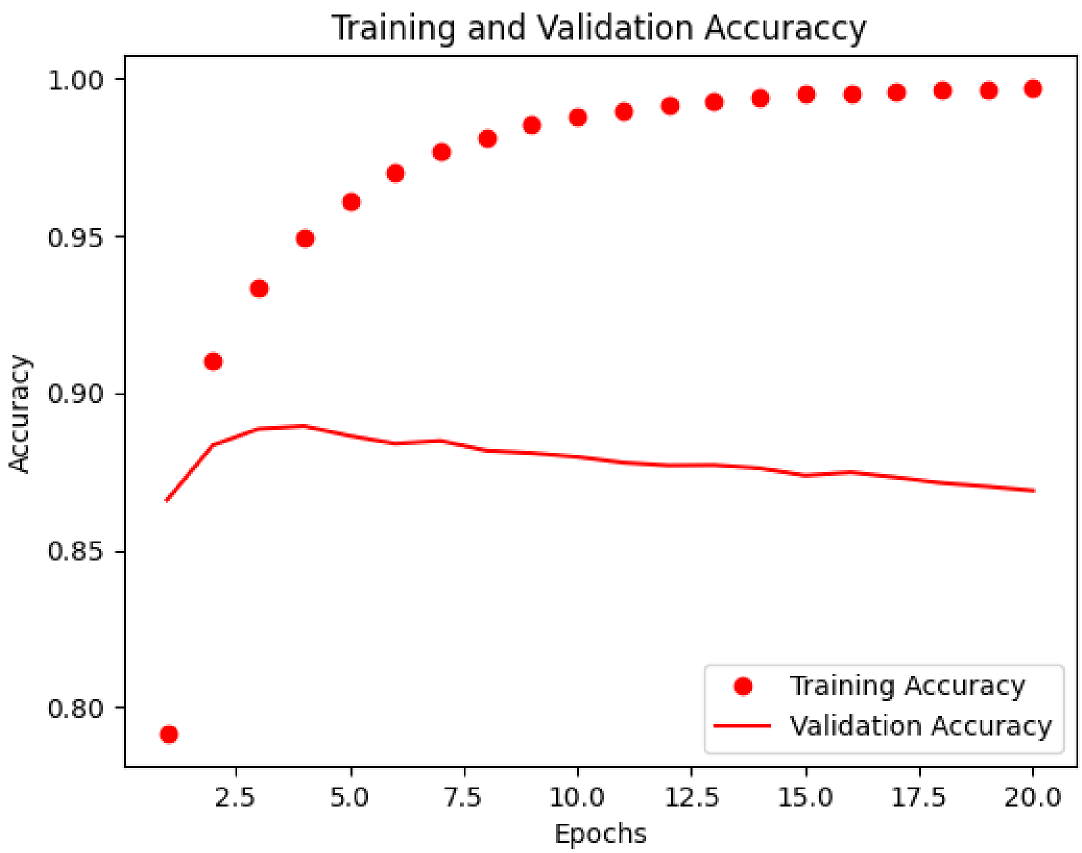
```
model.compile(optimizer='adam',
              loss = losses.mse,
              metrics = [metrics.binary_accuracy])



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 [==============================] - 1s 24ms/step - loss: 0.1636 -
binary_accuracy: 0.7918 - val_loss: 0.1163 - val_binary_accuracy: 0.8660
Epoch 2/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0871 -
binary_accuracy: 0.9104 - val_loss: 0.0966 - val_binary_accuracy: 0.8833
Epoch 3/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0658 -
binary_accuracy: 0.9332 - val_loss: 0.0889 - val_binary_accuracy: 0.8885
Epoch 4/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0529 -
binary_accuracy: 0.9493 - val_loss: 0.0858 - val_binary_accuracy: 0.8894
Epoch 5/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0439 -
binary_accuracy: 0.9609 - val_loss: 0.0847 - val_binary_accuracy: 0.8863
Epoch 6/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0371 -
binary_accuracy: 0.9697 - val_loss: 0.0852 - val_binary_accuracy: 0.8838
Epoch 7/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0316 -
binary_accuracy: 0.9765 - val_loss: 0.0850 - val_binary_accuracy: 0.8847
Epoch 8/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0272 -
binary_accuracy: 0.9809 - val_loss: 0.0860 - val_binary_accuracy: 0.8816
Epoch 9/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0235 -
binary_accuracy: 0.9853 - val_loss: 0.0869 - val_binary_accuracy: 0.8808
Epoch 10/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0204 -
binary_accuracy: 0.9879 - val_loss: 0.0882 - val_binary_accuracy: 0.8796
Epoch 11/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0179 -
binary_accuracy: 0.9896 - val_loss: 0.0895 - val_binary_accuracy: 0.8778
Epoch 12/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0157 -
```

```
binary_accuracy: 0.9915 - val_loss: 0.0906 - val_binary_accuracy: 0.8769
Epoch 13/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0140 -
binary_accuracy: 0.9929 - val_loss: 0.0915 - val_binary_accuracy: 0.8770
Epoch 14/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0124 -
binary_accuracy: 0.9941 - val_loss: 0.0926 - val_binary_accuracy: 0.8760
Epoch 15/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0111 -
binary_accuracy: 0.9948 - val_loss: 0.0937 - val_binary_accuracy: 0.8737
Epoch 16/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0099 -
binary_accuracy: 0.9953 - val_loss: 0.0949 - val_binary_accuracy: 0.8747
Epoch 17/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0089 -
binary_accuracy: 0.9959 - val_loss: 0.0961 - val_binary_accuracy: 0.8731
Epoch 18/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0080 -
binary_accuracy: 0.9963 - val_loss: 0.0972 - val_binary_accuracy: 0.8713
Epoch 19/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0072 -
binary_accuracy: 0.9965 - val_loss: 0.0979 - val_binary_accuracy: 0.8702
Epoch 20/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0066 -
binary_accuracy: 0.9969 - val_loss: 0.0989 - val_binary_accuracy: 0.8689
```


Training and Validation Loss
Training Loss
Validation Loss
Loss Value
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
Epochs
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0



[32]:

```
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=4, batch_size=512)
results = model.evaluate(x_test, y_test)

results
```

```
Epoch 1/4
49/49 [==============================] - 1s 7ms/step - loss: 0.1447 - accuracy:
0.8238
Epoch 2/4
49/49 [==============================] - 0s 6ms/step - loss: 0.0790 - accuracy:
0.9135
Epoch 3/4
```

```
49/49 [==============================] - 0s 7ms/step - loss: 0.0615 - accuracy:
0.9327
Epoch 4/4
49/49 [==============================] - 0s 6ms/step - loss: 0.0506 - accuracy:
0.9467
782/782 [==============================] - 26s 33ms/step - loss: 0.0867 -
accuracy: 0.8840
```

[32]:

```
[0.08667165786027908, 0.8840399980545044]
```

[33]:

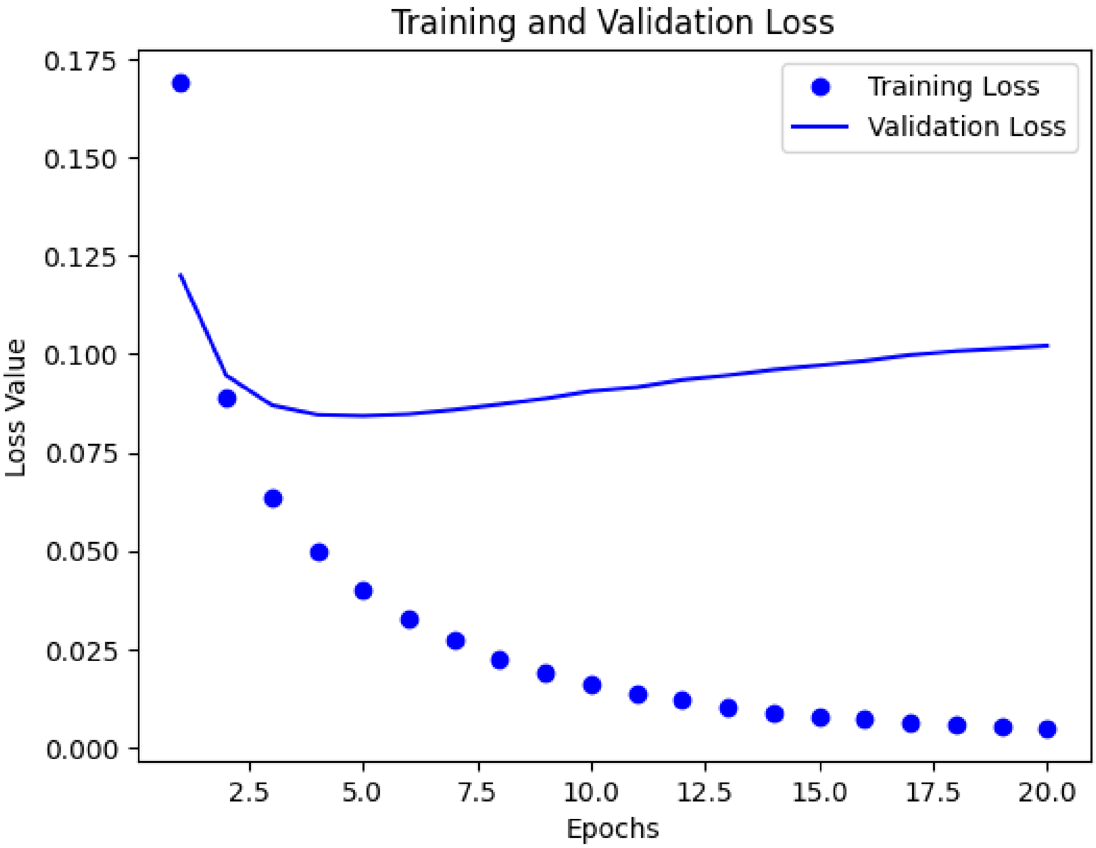
```
#implemented one hideden layer with 32 neurons and mse loss function

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))
```

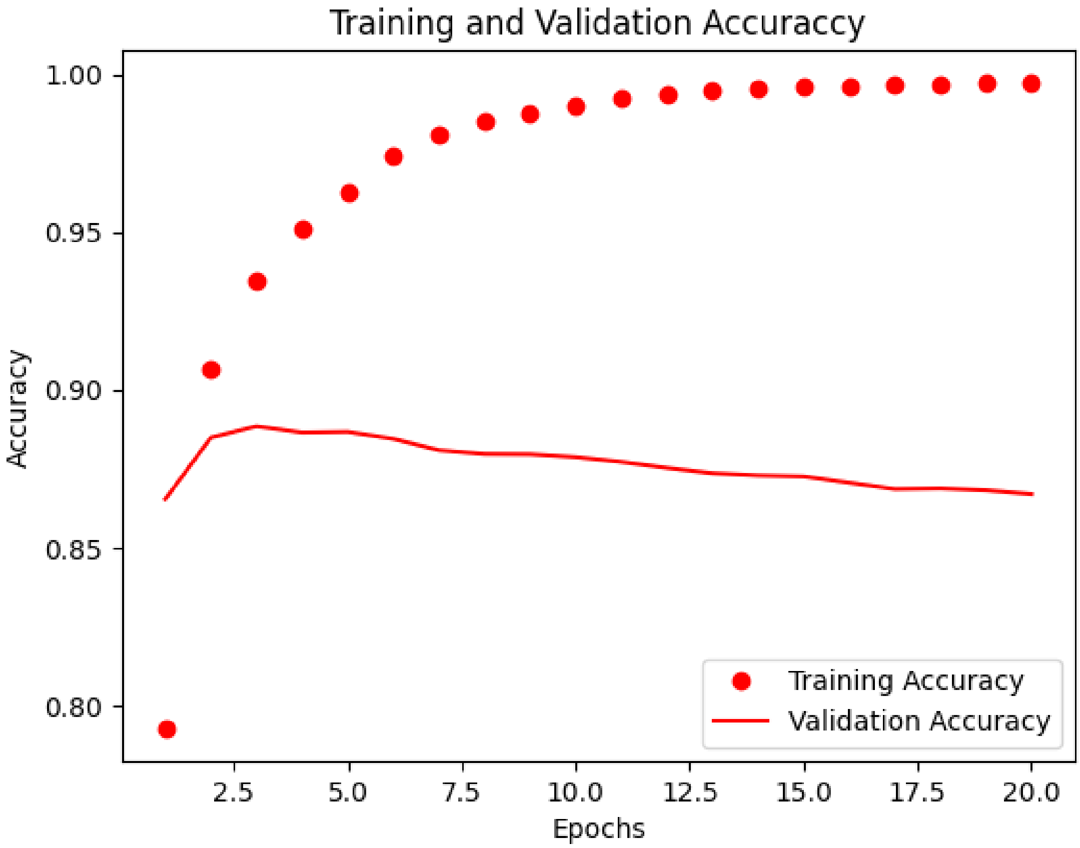
```python
history_dict = history.history
history_dict.keys()
# Plotting the training and validation loss

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: 0.1692 -
binary_accuracy: 0.7927 - val_loss: 0.1200 - val_binary_accuracy: 0.8653
Epoch 2/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0888 -
binary_accuracy: 0.9064 - val_loss: 0.0946 - val_binary_accuracy: 0.8848
```

```
Epoch 3/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0637 -
binary_accuracy: 0.9346 - val_loss: 0.0870 - val_binary_accuracy: 0.8884
Epoch 4/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0500 -
binary_accuracy: 0.9510 - val_loss: 0.0846 - val_binary_accuracy: 0.8864
Epoch 5/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0399 -
binary_accuracy: 0.9628 - val_loss: 0.0843 - val_binary_accuracy: 0.8866
Epoch 6/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0328 -
binary_accuracy: 0.9739 - val_loss: 0.0848 - val_binary_accuracy: 0.8844
Epoch 7/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0271 -
binary_accuracy: 0.9807 - val_loss: 0.0859 - val_binary_accuracy: 0.8808
Epoch 8/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0226 -
binary_accuracy: 0.9850 - val_loss: 0.0872 - val_binary_accuracy: 0.8796
Epoch 9/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0193 -
binary_accuracy: 0.9877 - val_loss: 0.0887 - val_binary_accuracy: 0.8795
Epoch 10/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0163 -
binary_accuracy: 0.9898 - val_loss: 0.0906 - val_binary_accuracy: 0.8786
Epoch 11/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0138 -
binary_accuracy: 0.9921 - val_loss: 0.0916 - val_binary_accuracy: 0.8771
Epoch 12/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0120 -
binary_accuracy: 0.9935 - val_loss: 0.0934 - val_binary_accuracy: 0.8752
Epoch 13/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0103 -
binary_accuracy: 0.9949 - val_loss: 0.0946 - val_binary_accuracy: 0.8735
Epoch 14/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0090 -
binary_accuracy: 0.9955 - val_loss: 0.0960 - val_binary_accuracy: 0.8728
Epoch 15/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0080 -
binary_accuracy: 0.9959 - val_loss: 0.0971 - val_binary_accuracy: 0.8725
Epoch 16/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0071 -
binary_accuracy: 0.9963 - val_loss: 0.0983 - val_binary_accuracy: 0.8705
Epoch 17/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0065 -
binary_accuracy: 0.9966 - val_loss: 0.0998 - val_binary_accuracy: 0.8685
Epoch 18/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0059 -
binary_accuracy: 0.9967 - val_loss: 0.1008 - val_binary_accuracy: 0.8687
```

```
Epoch 19/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0053 -
binary_accuracy: 0.9973 - val_loss: 0.1015 - val_binary_accuracy: 0.8681
Epoch 20/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0049 -
binary_accuracy: 0.9973 - val_loss: 0.1021 - val_binary_accuracy: 0.8669
```





[62]:
```
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=3, batch_size=512)
results = model.evaluate(x_test, y_test)

results
```

```
Epoch 1/3
49/49 [==============================] - 2s 13ms/step - loss: 0.1349 - accuracy:
0.8365
Epoch 2/3
49/49 [==============================] - 0s 10ms/step - loss: 0.0709 - accuracy:
0.9186
Epoch 3/3
```

```
49/49 [==============================] - 0s 8ms/step - loss: 0.0542 - accuracy:
0.9386
782/782 [==============================] - 25s 31ms/step - loss: 0.0862 -
accuracy: 0.8846
```

[62]:

```
[0.08622293174266815, 0.8845999836921692]
```

[35]:

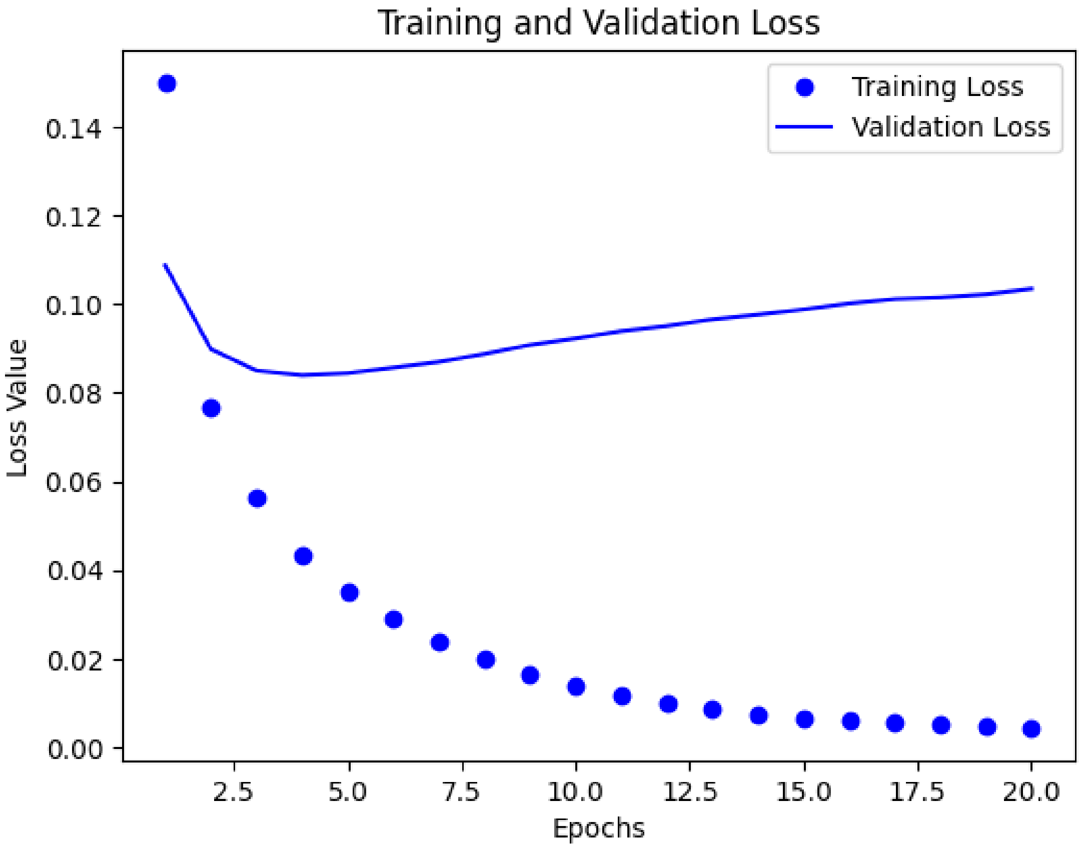
```python
#implemented one hideden layer with 32 neurons and mse loss function with dropout

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
```

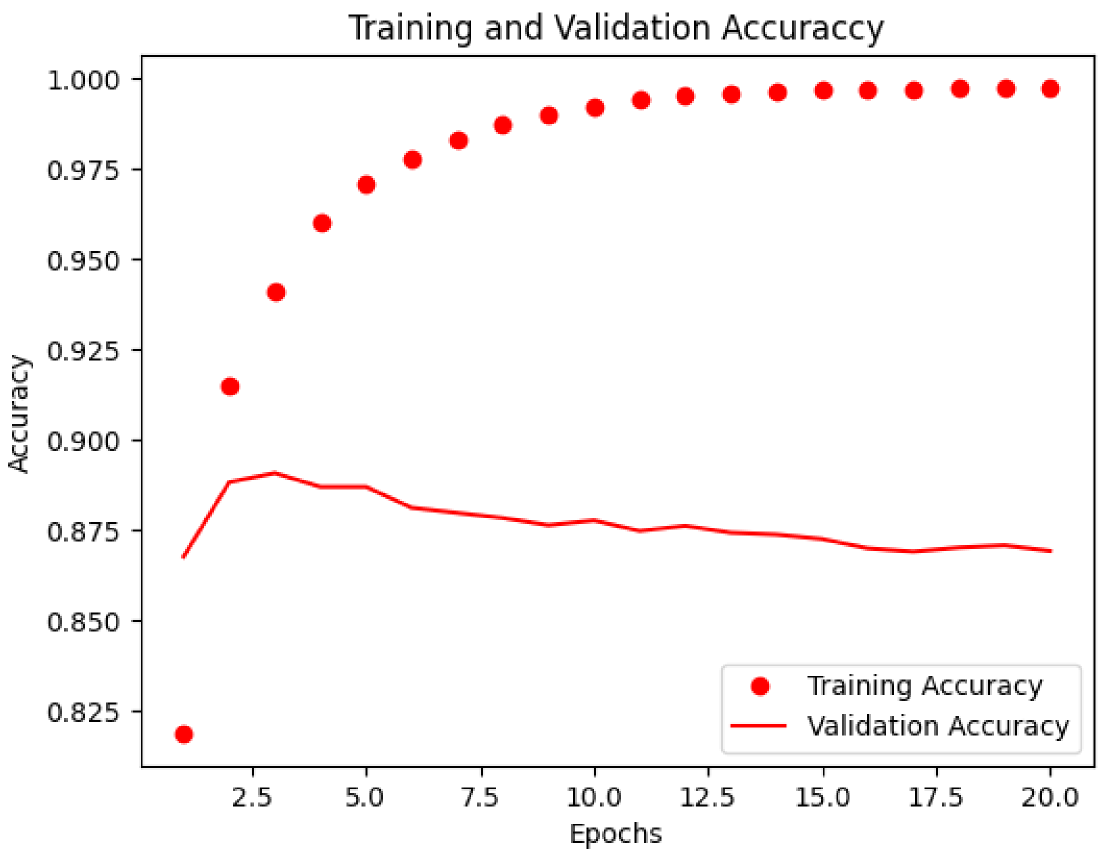
```python
history_dict.keys()
# Plotting the training and validation loss

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 23ms/step - loss: 0.1499 -
binary_accuracy: 0.8187 - val_loss: 0.1087 - val_binary_accuracy: 0.8676
Epoch 2/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0765 -
binary_accuracy: 0.9148 - val_loss: 0.0898 - val_binary_accuracy: 0.8882
Epoch 3/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0564 -
```

```
binary_accuracy: 0.9411 - val_loss: 0.0850 - val_binary_accuracy: 0.8907
Epoch 4/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0436 -
binary_accuracy: 0.9599 - val_loss: 0.0839 - val_binary_accuracy: 0.8869
Epoch 5/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0351 -
binary_accuracy: 0.9705 - val_loss: 0.0844 - val_binary_accuracy: 0.8869
Epoch 6/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0289 -
binary_accuracy: 0.9779 - val_loss: 0.0857 - val_binary_accuracy: 0.8811
Epoch 7/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0238 -
binary_accuracy: 0.9829 - val_loss: 0.0869 - val_binary_accuracy: 0.8797
Epoch 8/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0200 -
binary_accuracy: 0.9870 - val_loss: 0.0887 - val_binary_accuracy: 0.8783
Epoch 9/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0166 -
binary_accuracy: 0.9899 - val_loss: 0.0908 - val_binary_accuracy: 0.8763
Epoch 10/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0138 -
binary_accuracy: 0.9921 - val_loss: 0.0922 - val_binary_accuracy: 0.8776
Epoch 11/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0117 -
binary_accuracy: 0.9939 - val_loss: 0.0939 - val_binary_accuracy: 0.8747
Epoch 12/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0100 -
binary_accuracy: 0.9951 - val_loss: 0.0950 - val_binary_accuracy: 0.8761
Epoch 13/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0087 -
binary_accuracy: 0.9955 - val_loss: 0.0965 - val_binary_accuracy: 0.8742
Epoch 14/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0075 -
binary_accuracy: 0.9962 - val_loss: 0.0976 - val_binary_accuracy: 0.8737
Epoch 15/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0067 -
binary_accuracy: 0.9968 - val_loss: 0.0988 - val_binary_accuracy: 0.8725
Epoch 16/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0061 -
binary_accuracy: 0.9969 - val_loss: 0.1001 - val_binary_accuracy: 0.8699
Epoch 17/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0056 -
binary_accuracy: 0.9970 - val_loss: 0.1011 - val_binary_accuracy: 0.8690
Epoch 18/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0052 -
binary_accuracy: 0.9971 - val_loss: 0.1015 - val_binary_accuracy: 0.8701
Epoch 19/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0047 -
```

```
binary_accuracy: 0.9973 - val_loss: 0.1022 - val_binary_accuracy: 0.8707
Epoch 20/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0044 -
binary_accuracy: 0.9973 - val_loss: 0.1034 - val_binary_accuracy: 0.8692
```





[63]:

```
model = models.Sequential()
model.add(layers.Dense(32, 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=3, batch_size=512)
results = model.evaluate(x_test, y_test)

results
```

```
Epoch 1/3
49/49 [==============================] - 1s 12ms/step - loss: 0.1295 - accuracy:
0.8341
Epoch 2/3
49/49 [==============================] - 0s 8ms/step - loss: 0.0679 - accuracy:
0.9211
```

```
Epoch 3/3
49/49 [==============================] - 0s 9ms/step - loss: 0.0513 - accuracy:
0.9426
782/782 [==============================] - 25s 32ms/step - loss: 0.0868 -
accuracy: 0.8834
```

[63]:

```
[0.08677373826503754, 0.8833600282669067]
```

[37]:

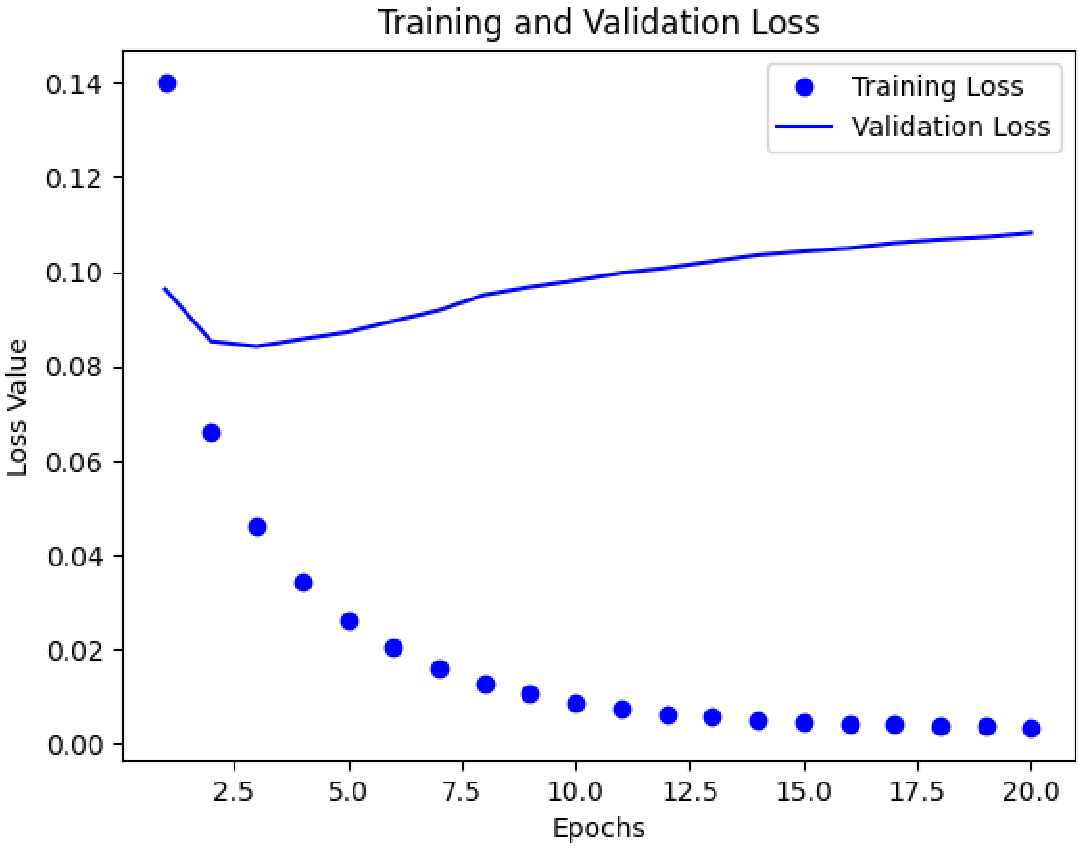
```
#implemented one hideden layer with 64 neurons and mse loss function

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
```

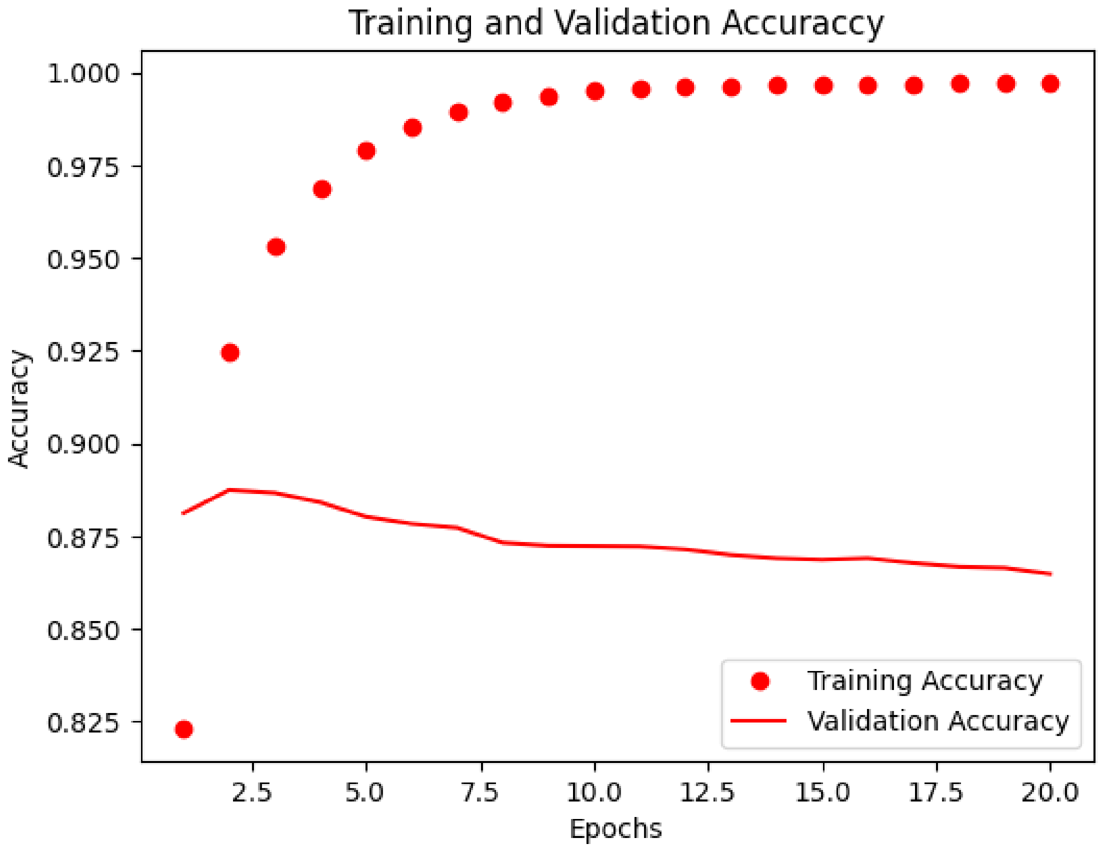
```
history_dict.keys()
# Plotting the training and validation loss

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 [==============================] - 2s 39ms/step - loss: 0.1401 -
binary_accuracy: 0.8231 - val_loss: 0.0963 - val_binary_accuracy: 0.8812
Epoch 2/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0659 -
binary_accuracy: 0.9247 - val_loss: 0.0853 - val_binary_accuracy: 0.8874
Epoch 3/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0461 -
```

```
binary_accuracy: 0.9533 - val_loss: 0.0841 - val_binary_accuracy: 0.8866
Epoch 4/20
30/30 [==============================] - 0s 17ms/step - loss: 0.0343 -
binary_accuracy: 0.9690 - val_loss: 0.0857 - val_binary_accuracy: 0.8841
Epoch 5/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0263 -
binary_accuracy: 0.9794 - val_loss: 0.0872 - val_binary_accuracy: 0.8802
Epoch 6/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0204 -
binary_accuracy: 0.9855 - val_loss: 0.0895 - val_binary_accuracy: 0.8783
Epoch 7/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0161 -
binary_accuracy: 0.9893 - val_loss: 0.0918 - val_binary_accuracy: 0.8772
Epoch 8/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0130 -
binary_accuracy: 0.9923 - val_loss: 0.0951 - val_binary_accuracy: 0.8732
Epoch 9/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0106 -
binary_accuracy: 0.9936 - val_loss: 0.0967 - val_binary_accuracy: 0.8724
Epoch 10/20
30/30 [==============================] - 0s 17ms/step - loss: 0.0087 -
binary_accuracy: 0.9950 - val_loss: 0.0981 - val_binary_accuracy: 0.8723
Epoch 11/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0074 -
binary_accuracy: 0.9956 - val_loss: 0.0997 - val_binary_accuracy: 0.8722
Epoch 12/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0064 -
binary_accuracy: 0.9961 - val_loss: 0.1008 - val_binary_accuracy: 0.8714
Epoch 13/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0057 -
binary_accuracy: 0.9963 - val_loss: 0.1021 - val_binary_accuracy: 0.8699
Epoch 14/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0051 -
binary_accuracy: 0.9968 - val_loss: 0.1035 - val_binary_accuracy: 0.8690
Epoch 15/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0047 -
binary_accuracy: 0.9968 - val_loss: 0.1043 - val_binary_accuracy: 0.8686
Epoch 16/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0043 -
binary_accuracy: 0.9968 - val_loss: 0.1049 - val_binary_accuracy: 0.8690
Epoch 17/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0040 -
binary_accuracy: 0.9970 - val_loss: 0.1060 - val_binary_accuracy: 0.8677
Epoch 18/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0038 -
binary_accuracy: 0.9971 - val_loss: 0.1068 - val_binary_accuracy: 0.8667
Epoch 19/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0036 -
```

```
binary_accuracy: 0.9972 - val_loss: 0.1073 - val_binary_accuracy: 0.8664
Epoch 20/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0034 -
binary_accuracy: 0.9973 - val_loss: 0.1082 - val_binary_accuracy: 0.8648
```





[38]:

```
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=2, batch_size=512)
results = model.evaluate(x_test, y_test)

results
```

```
Epoch 1/2
49/49 [==============================] - 1s 11ms/step - loss: 0.1244 - accuracy:
0.8376
Epoch 2/2
49/49 [==============================] - 0s 9ms/step - loss: 0.0610 - accuracy:
0.9288
782/782 [==============================] - 22s 28ms/step - loss: 0.0858 -
accuracy: 0.8852
```

[38]:

```
[0.0858103409409523, 0.8851600289344788]
```

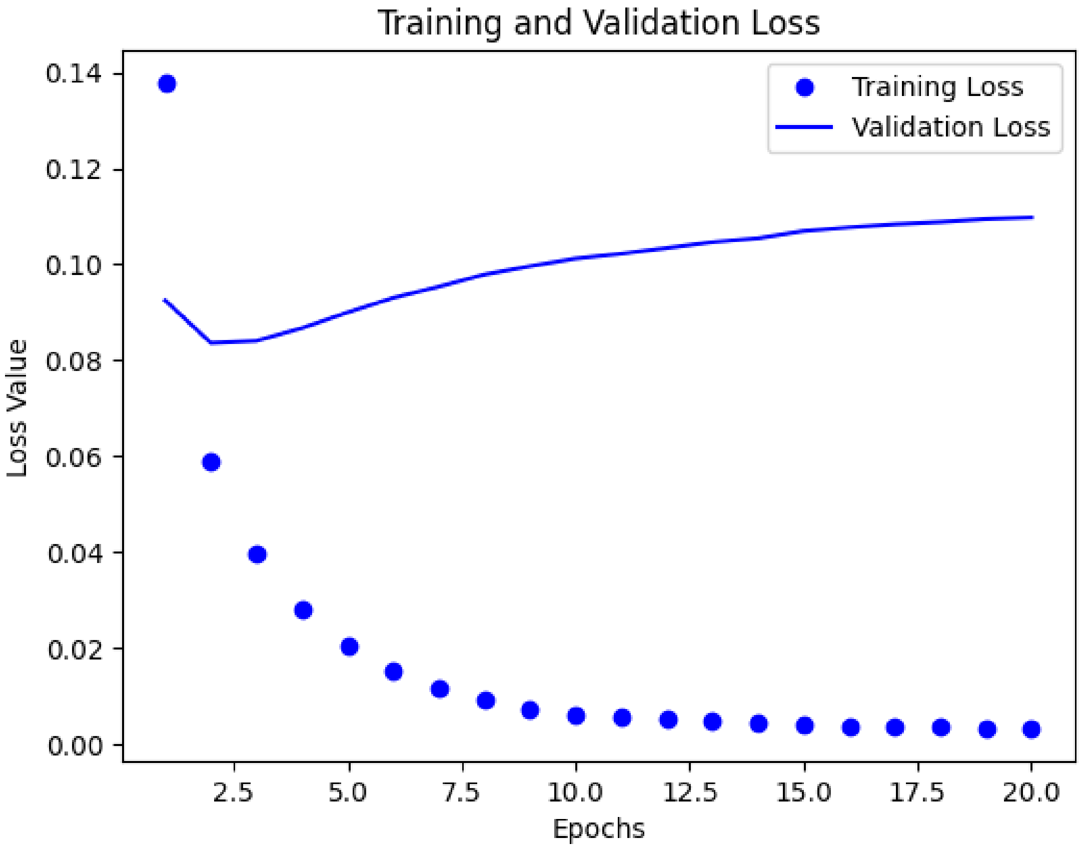
[39]:

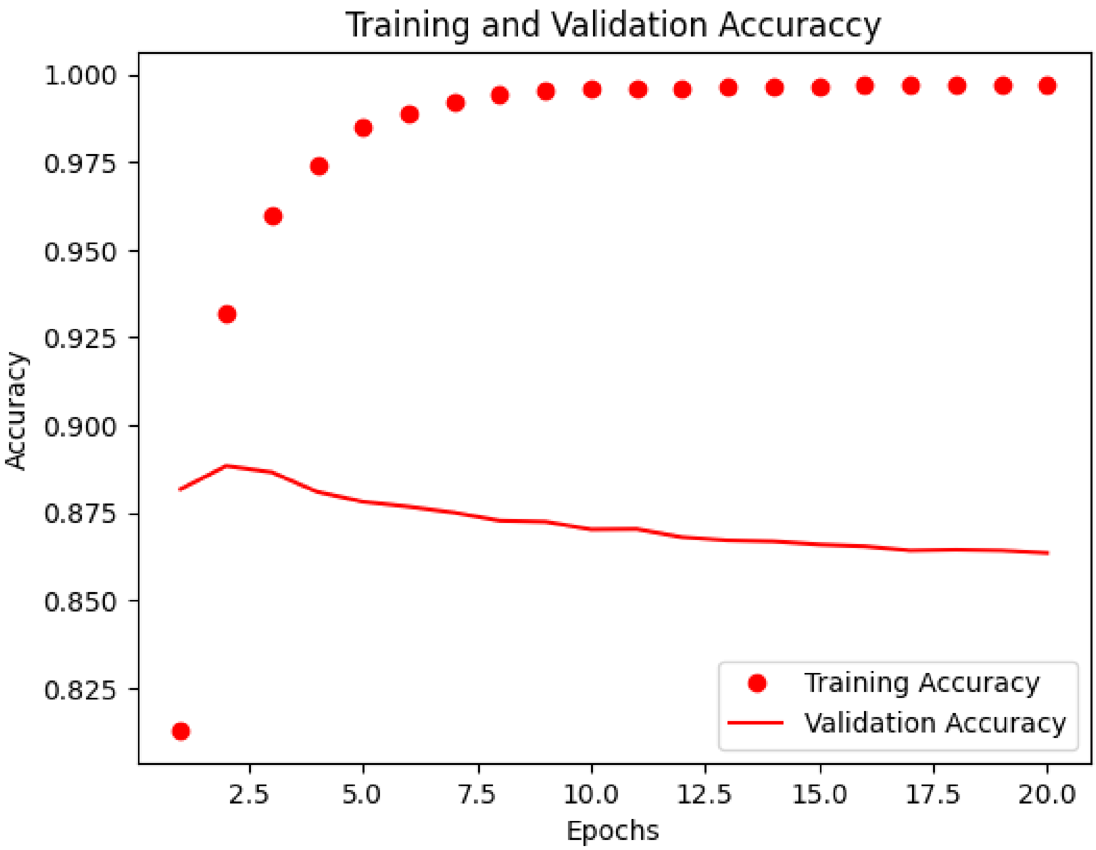
```python
#implemented one hideden layer with 128 neurons and mse loss function



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 31ms/step - loss: 0.1379 -
binary_accuracy: 0.8129 - val_loss: 0.0925 - val_binary_accuracy: 0.8817
Epoch 2/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0590 -
binary_accuracy: 0.9318 - val_loss: 0.0836 - val_binary_accuracy: 0.8883
Epoch 3/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0397 -
binary_accuracy: 0.9595 - val_loss: 0.0841 - val_binary_accuracy: 0.8865
Epoch 4/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0283 -
binary_accuracy: 0.9743 - val_loss: 0.0867 - val_binary_accuracy: 0.8809
Epoch 5/20
```

```
30/30 [==============================] - 1s 20ms/step - loss: 0.0205 -
binary_accuracy: 0.9848 - val_loss: 0.0900 - val_binary_accuracy: 0.8781
Epoch 6/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0154 -
binary_accuracy: 0.9890 - val_loss: 0.0930 - val_binary_accuracy: 0.8767
Epoch 7/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0118 -
binary_accuracy: 0.9923 - val_loss: 0.0954 - val_binary_accuracy: 0.8750
Epoch 8/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0092 -
binary_accuracy: 0.9944 - val_loss: 0.0979 - val_binary_accuracy: 0.8727
Epoch 9/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0074 -
binary_accuracy: 0.9955 - val_loss: 0.0996 - val_binary_accuracy: 0.8724
Epoch 10/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0063 -
binary_accuracy: 0.9958 - val_loss: 0.1012 - val_binary_accuracy: 0.8702
Epoch 11/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0056 -
binary_accuracy: 0.9960 - val_loss: 0.1022 - val_binary_accuracy: 0.8703
Epoch 12/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0051 -
binary_accuracy: 0.9961 - val_loss: 0.1034 - val_binary_accuracy: 0.8680
Epoch 13/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0047 -
binary_accuracy: 0.9963 - val_loss: 0.1046 - val_binary_accuracy: 0.8671
Epoch 14/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0044 -
binary_accuracy: 0.9964 - val_loss: 0.1054 - val_binary_accuracy: 0.8668
Epoch 15/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0041 -
binary_accuracy: 0.9967 - val_loss: 0.1069 - val_binary_accuracy: 0.8659
Epoch 16/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0038 -
binary_accuracy: 0.9969 - val_loss: 0.1077 - val_binary_accuracy: 0.8654
Epoch 17/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0036 -
binary_accuracy: 0.9969 - val_loss: 0.1083 - val_binary_accuracy: 0.8642
Epoch 18/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0035 -
binary_accuracy: 0.9971 - val_loss: 0.1088 - val_binary_accuracy: 0.8644
Epoch 19/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0034 -
binary_accuracy: 0.9971 - val_loss: 0.1094 - val_binary_accuracy: 0.8642
Epoch 20/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0033 -
binary_accuracy: 0.9971 - val_loss: 0.1098 - val_binary_accuracy: 0.8635
```


Training and Validation Loss
Training Loss
Validation Loss
Loss Value
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
Epochs
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0



[40]:

```
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=2, batch_size=512)
results = model.evaluate(x_test, y_test)

results
```

```
Epoch 1/2
49/49 [==============================] - 1s 14ms/step - loss: 0.1132 - accuracy:
0.8465
Epoch 2/2
49/49 [==============================] - 1s 14ms/step - loss: 0.0553 - accuracy:
0.9316
782/782 [==============================] - 3s 4ms/step - loss: 0.0882 -
accuracy: 0.8803
```

[40]:

```
[0.088201604783535, 0.8803200125694275]
```

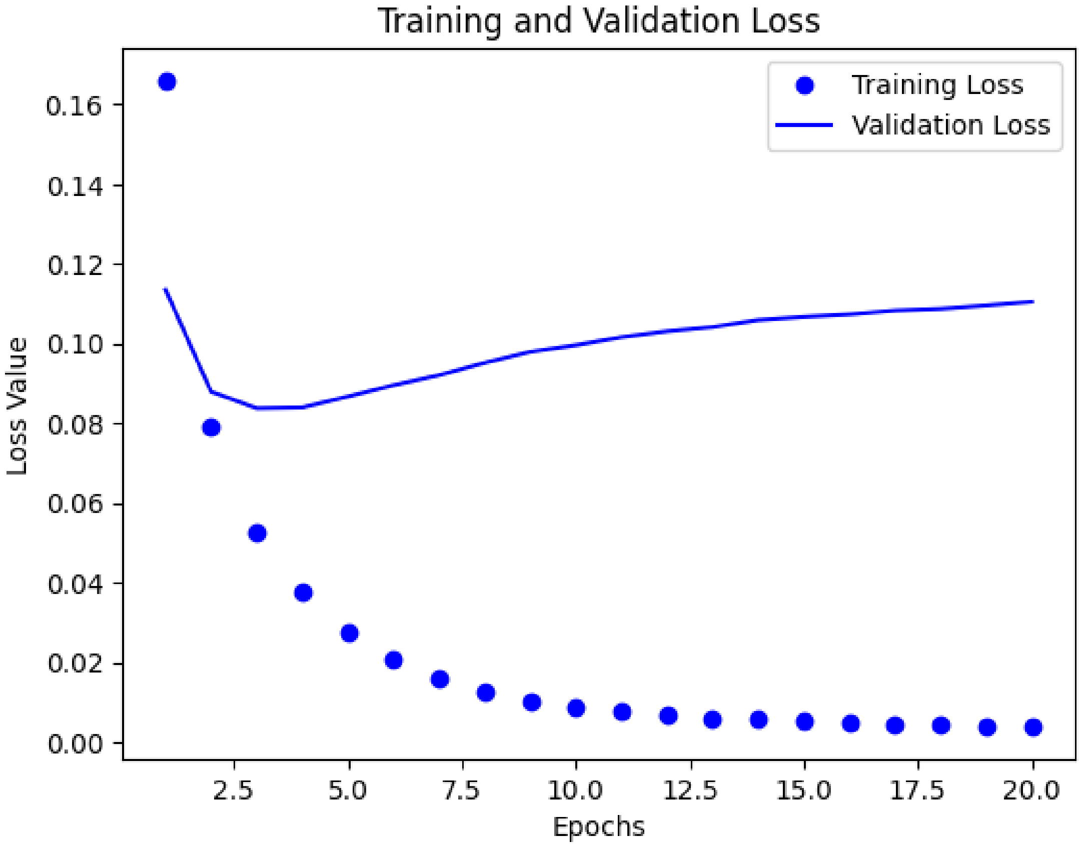
[41]:

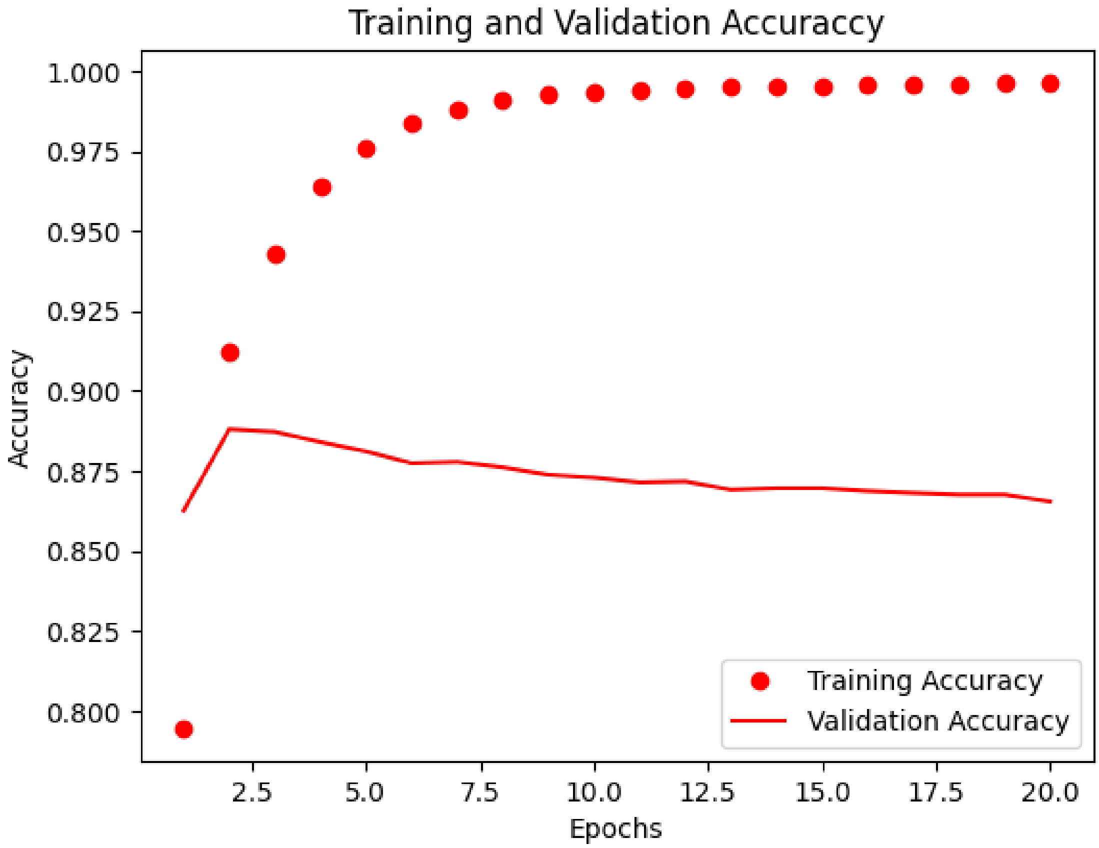
```
#implemented two hidden layer with 16 neurons and mse loss function



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
```

```python
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 25ms/step - loss: 0.1660 -
binary_accuracy: 0.7945 - val_loss: 0.1135 - val_binary_accuracy: 0.8626
Epoch 2/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0792 -
binary_accuracy: 0.9121 - val_loss: 0.0879 - val_binary_accuracy: 0.8880
Epoch 3/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0528 -
binary_accuracy: 0.9426 - val_loss: 0.0838 - val_binary_accuracy: 0.8872
Epoch 4/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0378 -
binary_accuracy: 0.9639 - val_loss: 0.0840 - val_binary_accuracy: 0.8840
Epoch 5/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0276 -
binary_accuracy: 0.9757 - val_loss: 0.0867 - val_binary_accuracy: 0.8811
Epoch 6/20
```

```
30/30 [==============================] - 0s 12ms/step - loss: 0.0206 -
binary_accuracy: 0.9837 - val_loss: 0.0895 - val_binary_accuracy: 0.8774
Epoch 7/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0157 -
binary_accuracy: 0.9879 - val_loss: 0.0920 - val_binary_accuracy: 0.8778
Epoch 8/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0124 -
binary_accuracy: 0.9909 - val_loss: 0.0952 - val_binary_accuracy: 0.8761
Epoch 9/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0101 -
binary_accuracy: 0.9926 - val_loss: 0.0979 - val_binary_accuracy: 0.8738
Epoch 10/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0087 -
binary_accuracy: 0.9933 - val_loss: 0.0996 - val_binary_accuracy: 0.8729
Epoch 11/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0075 -
binary_accuracy: 0.9940 - val_loss: 0.1016 - val_binary_accuracy: 0.8714
Epoch 12/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0067 -
binary_accuracy: 0.9945 - val_loss: 0.1031 - val_binary_accuracy: 0.8717
Epoch 13/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0060 -
binary_accuracy: 0.9949 - val_loss: 0.1041 - val_binary_accuracy: 0.8691
Epoch 14/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0057 -
binary_accuracy: 0.9952 - val_loss: 0.1059 - val_binary_accuracy: 0.8696
Epoch 15/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0053 -
binary_accuracy: 0.9954 - val_loss: 0.1067 - val_binary_accuracy: 0.8696
Epoch 16/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0049 -
binary_accuracy: 0.9957 - val_loss: 0.1073 - val_binary_accuracy: 0.8687
Epoch 17/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0046 -
binary_accuracy: 0.9959 - val_loss: 0.1083 - val_binary_accuracy: 0.8681
Epoch 18/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0044 -
binary_accuracy: 0.9960 - val_loss: 0.1087 - val_binary_accuracy: 0.8676
Epoch 19/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0041 -
binary_accuracy: 0.9962 - val_loss: 0.1096 - val_binary_accuracy: 0.8676
Epoch 20/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0039 -
binary_accuracy: 0.9964 - val_loss: 0.1105 - val_binary_accuracy: 0.8654
```


Training and Validation Loss
● Training Loss
Validation Loss
Loss Value
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epochs



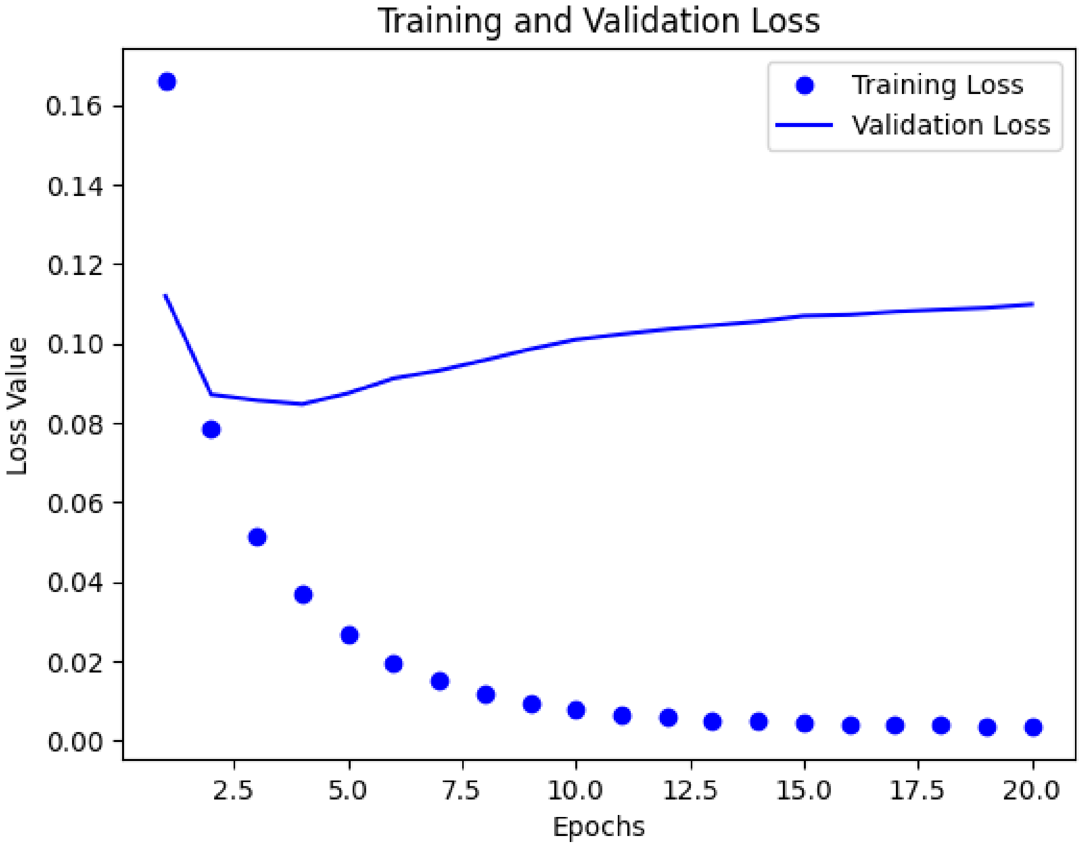
[42]:

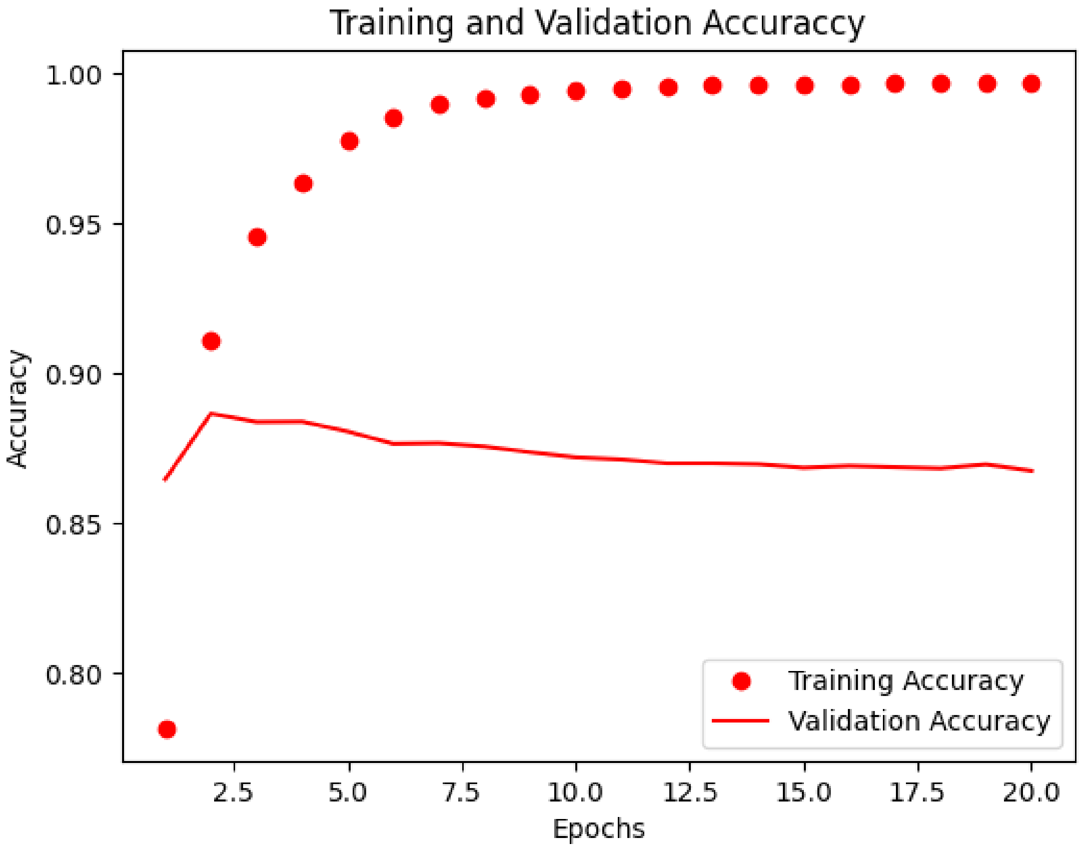
```
#implemented two hideden layer with 16 neurons and mse loss function



from keras import models
from tensorflow.keras import layers

model = models.Sequential()
model.add(layers.Dense(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 [==============================] - 1s 24ms/step - loss: 0.1662 -
binary_accuracy: 0.7814 - val_loss: 0.1119 - val_binary_accuracy: 0.8645
Epoch 2/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0785 -
binary_accuracy: 0.9109 - val_loss: 0.0871 - val_binary_accuracy: 0.8864
Epoch 3/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0513 -
binary_accuracy: 0.9455 - val_loss: 0.0856 - val_binary_accuracy: 0.8836
Epoch 4/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0368 -
binary_accuracy: 0.9634 - val_loss: 0.0847 - val_binary_accuracy: 0.8837
Epoch 5/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0265 -
binary_accuracy: 0.9773 - val_loss: 0.0874 - val_binary_accuracy: 0.8804
Epoch 6/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0195 -
binary_accuracy: 0.9851 - val_loss: 0.0911 - val_binary_accuracy: 0.8763
Epoch 7/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0149 -
binary_accuracy: 0.9899 - val_loss: 0.0931 - val_binary_accuracy: 0.8765
Epoch 8/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0117 -
binary_accuracy: 0.9917 - val_loss: 0.0958 - val_binary_accuracy: 0.8754
Epoch 9/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0094 -
binary_accuracy: 0.9931 - val_loss: 0.0985 - val_binary_accuracy: 0.8735
Epoch 10/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0076 -
binary_accuracy: 0.9943 - val_loss: 0.1009 - val_binary_accuracy: 0.8718
Epoch 11/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0065 -
binary_accuracy: 0.9950 - val_loss: 0.1023 - val_binary_accuracy: 0.8711
Epoch 12/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0058 -
binary_accuracy: 0.9954 - val_loss: 0.1035 - val_binary_accuracy: 0.8698
Epoch 13/20
```

```
30/30 [==============================] - 0s 12ms/step - loss: 0.0052 -
binary_accuracy: 0.9959 - val_loss: 0.1045 - val_binary_accuracy: 0.8698
Epoch 14/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0048 -
binary_accuracy: 0.9961 - val_loss: 0.1055 - val_binary_accuracy: 0.8695
Epoch 15/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0044 -
binary_accuracy: 0.9963 - val_loss: 0.1069 - val_binary_accuracy: 0.8683
Epoch 16/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0042 -
binary_accuracy: 0.9964 - val_loss: 0.1072 - val_binary_accuracy: 0.8689
Epoch 17/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0040 -
binary_accuracy: 0.9965 - val_loss: 0.1080 - val_binary_accuracy: 0.8685
Epoch 18/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0038 -
binary_accuracy: 0.9966 - val_loss: 0.1085 - val_binary_accuracy: 0.8681
Epoch 19/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0037 -
binary_accuracy: 0.9967 - val_loss: 0.1089 - val_binary_accuracy: 0.8694
Epoch 20/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0035 -
binary_accuracy: 0.9968 - val_loss: 0.1098 - val_binary_accuracy: 0.8673
```


Training and Validation Loss
Training Loss
Validation Loss
Loss Value
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epochs



[43]:

```
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=4, batch_size=512)
results = model.evaluate(x_test, y_test)

results
```

```
Epoch 1/4
49/49 [==============================] - 1s 10ms/step - loss: 0.1382 - accuracy:
0.8250
Epoch 2/4
49/49 [==============================] - 0s 9ms/step - loss: 0.0657 - accuracy:
0.9208
Epoch 3/4
```

```
49/49 [==============================] - 0s 10ms/step - loss: 0.0468 - accuracy:
0.9465
Epoch 4/4
49/49 [==============================] - 0s 8ms/step - loss: 0.0364 - accuracy:
0.9600
782/782 [==============================] - 22s 28ms/step - loss: 0.0922 -
accuracy: 0.8752
```

[43]:

```
[0.09222698211669922, 0.8751999735832214]
```

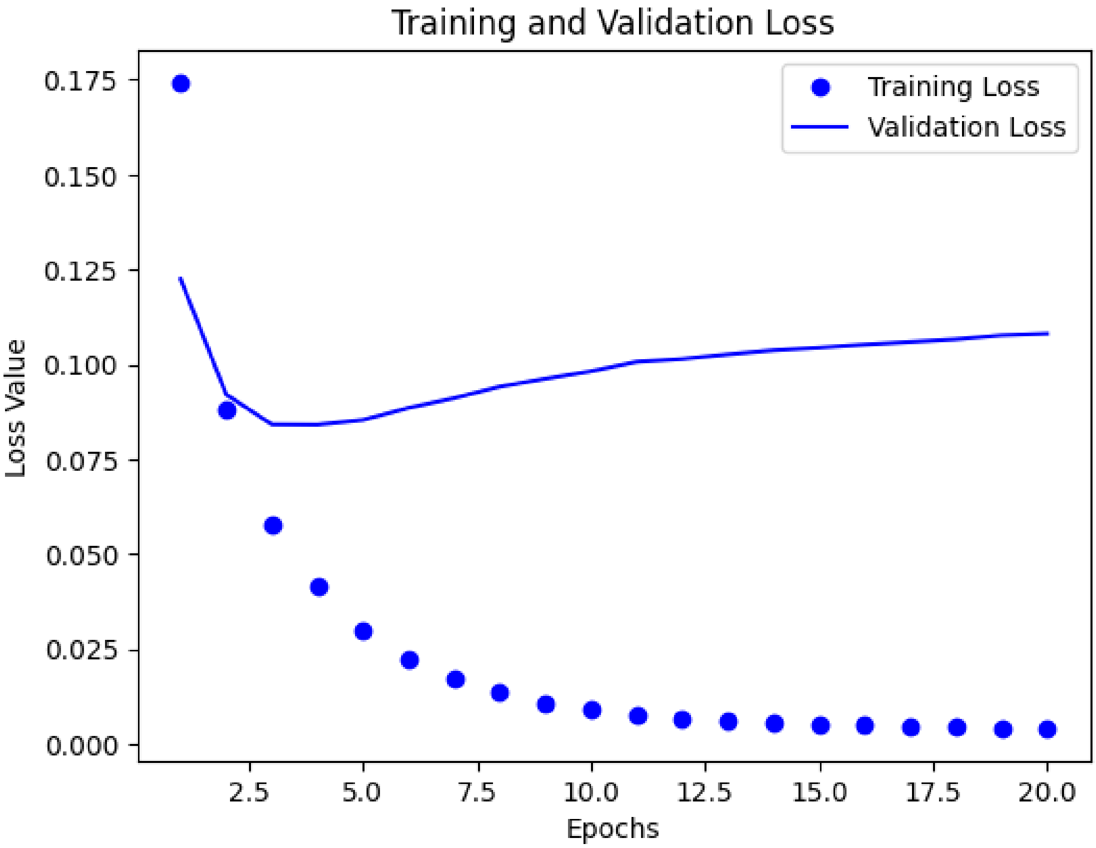
[44]:

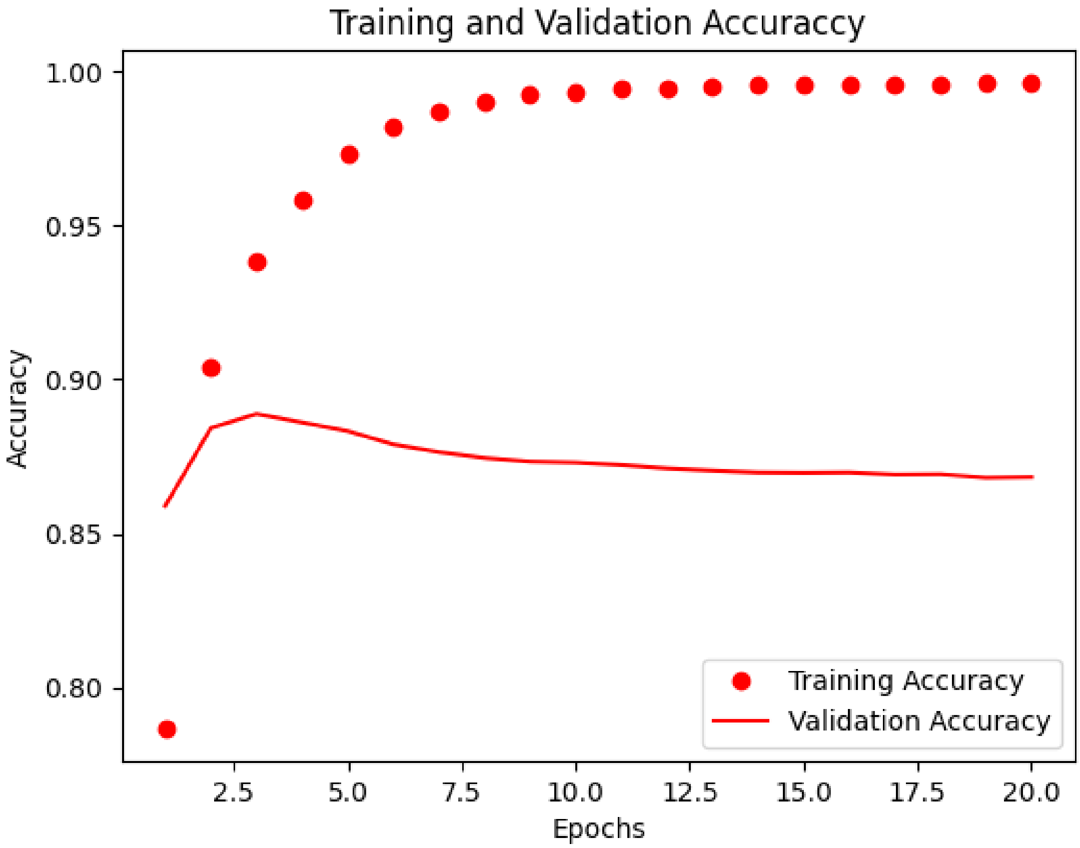
```python
#implemented two hideden layer with 16 neurons and mse loss function with dropout

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 [==============================] - 1s 22ms/step - loss: 0.1743 -
binary_accuracy: 0.7868 - val_loss: 0.1225 - val_binary_accuracy: 0.8590
Epoch 2/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0883 -
binary_accuracy: 0.9037 - val_loss: 0.0921 - val_binary_accuracy: 0.8842
Epoch 3/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0577 -
```

```
binary_accuracy: 0.9381 - val_loss: 0.0842 - val_binary_accuracy: 0.8888
Epoch 4/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0414 -
binary_accuracy: 0.9581 - val_loss: 0.0842 - val_binary_accuracy: 0.8860
Epoch 5/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0300 -
binary_accuracy: 0.9733 - val_loss: 0.0854 - val_binary_accuracy: 0.8832
Epoch 6/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0224 -
binary_accuracy: 0.9821 - val_loss: 0.0886 - val_binary_accuracy: 0.8789
Epoch 7/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0173 -
binary_accuracy: 0.9870 - val_loss: 0.0911 - val_binary_accuracy: 0.8764
Epoch 8/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0135 -
binary_accuracy: 0.9900 - val_loss: 0.0942 - val_binary_accuracy: 0.8745
Epoch 9/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0107 -
binary_accuracy: 0.9923 - val_loss: 0.0962 - val_binary_accuracy: 0.8733
Epoch 10/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0091 -
binary_accuracy: 0.9932 - val_loss: 0.0982 - val_binary_accuracy: 0.8730
Epoch 11/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0076 -
binary_accuracy: 0.9943 - val_loss: 0.1007 - val_binary_accuracy: 0.8722
Epoch 12/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0068 -
binary_accuracy: 0.9945 - val_loss: 0.1014 - val_binary_accuracy: 0.8711
Epoch 13/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0061 -
binary_accuracy: 0.9952 - val_loss: 0.1026 - val_binary_accuracy: 0.8704
Epoch 14/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0056 -
binary_accuracy: 0.9953 - val_loss: 0.1037 - val_binary_accuracy: 0.8698
Epoch 15/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0052 -
binary_accuracy: 0.9957 - val_loss: 0.1044 - val_binary_accuracy: 0.8697
Epoch 16/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0049 -
binary_accuracy: 0.9958 - val_loss: 0.1052 - val_binary_accuracy: 0.8698
Epoch 17/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0047 -
binary_accuracy: 0.9959 - val_loss: 0.1059 - val_binary_accuracy: 0.8691
Epoch 18/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0046 -
binary_accuracy: 0.9959 - val_loss: 0.1066 - val_binary_accuracy: 0.8692
Epoch 19/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0043 -
```

```
binary_accuracy: 0.9961 - val_loss: 0.1077 - val_binary_accuracy: 0.8681
Epoch 20/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0042 -
binary_accuracy: 0.9963 - val_loss: 0.1081 - val_binary_accuracy: 0.8683
```





[45]:

```
model = models.Sequential()
model.add(layers.Dense(16, activation='tanh', input_shape=(10000,)))
layers.Dropout(0.5),
model.add(layers.Dense(16, 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 [==============================] - 1s 7ms/step - loss: 0.1542 - accuracy:
0.8224
Epoch 2/4
49/49 [==============================] - 0s 6ms/step - loss: 0.0739 - accuracy:
```

```
0.9134
Epoch 3/4
49/49 [==============================] - 0s 6ms/step - loss: 0.0518 - accuracy:
0.9410
Epoch 4/4
49/49 [==============================] - 0s 6ms/step - loss: 0.0398 - accuracy:
0.9578
782/782 [==============================] - 15s 19ms/step - loss: 0.0884 -
accuracy: 0.8811
```

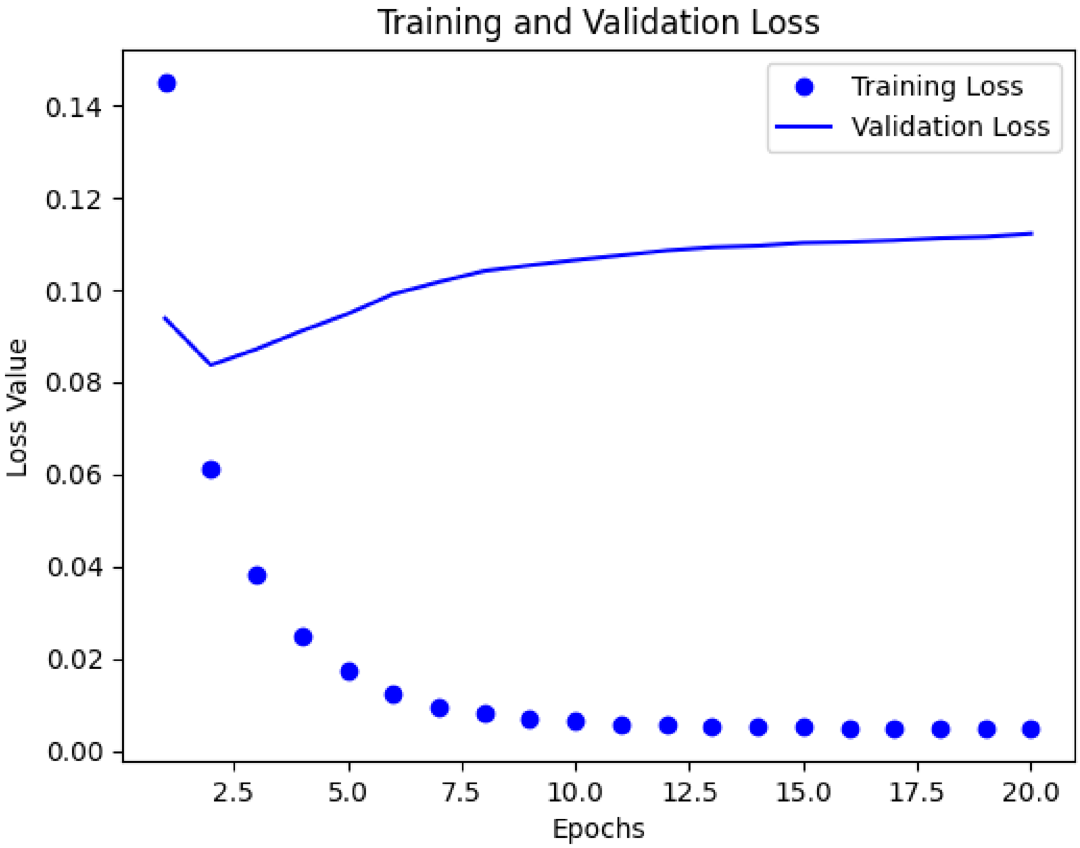
```
[45]: [0.08842990547418594, 0.8810799717903137]
```

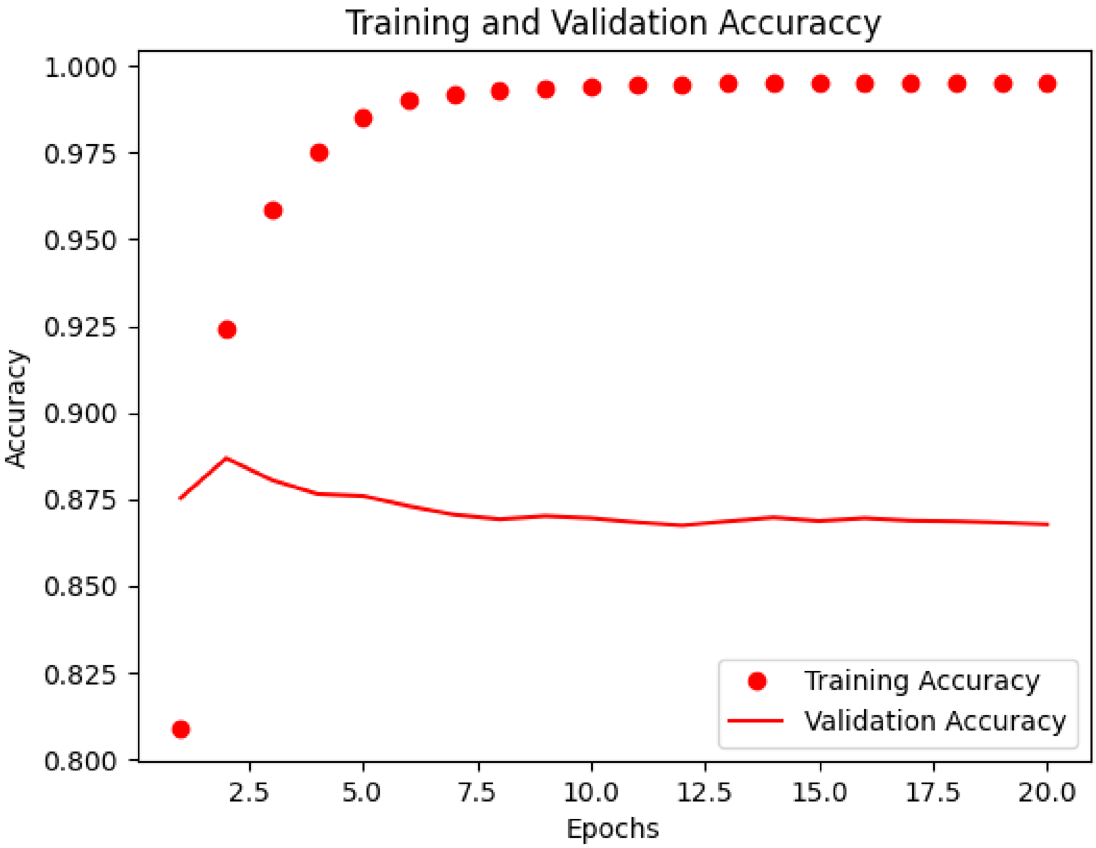
```
[46]: #implemented two hidden layer with 32 neurons and mse loss function



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(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 27ms/step - loss: 0.1450 -
binary_accuracy: 0.8089 - val_loss: 0.0937 - val_binary_accuracy: 0.8753
Epoch 2/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0609 -
binary_accuracy: 0.9239 - val_loss: 0.0836 - val_binary_accuracy: 0.8868
```

```
Epoch 3/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0380 -
binary_accuracy: 0.9583 - val_loss: 0.0871 - val_binary_accuracy: 0.8805
Epoch 4/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0249 -
binary_accuracy: 0.9754 - val_loss: 0.0911 - val_binary_accuracy: 0.8765
Epoch 5/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0173 -
binary_accuracy: 0.9853 - val_loss: 0.0947 - val_binary_accuracy: 0.8759
Epoch 6/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0124 -
binary_accuracy: 0.9901 - val_loss: 0.0990 - val_binary_accuracy: 0.8730
Epoch 7/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0096 -
binary_accuracy: 0.9919 - val_loss: 0.1016 - val_binary_accuracy: 0.8705
Epoch 8/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0080 -
binary_accuracy: 0.9930 - val_loss: 0.1040 - val_binary_accuracy: 0.8692
Epoch 9/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0071 -
binary_accuracy: 0.9937 - val_loss: 0.1053 - val_binary_accuracy: 0.8701
Epoch 10/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0063 -
binary_accuracy: 0.9942 - val_loss: 0.1064 - val_binary_accuracy: 0.8695
Epoch 11/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0058 -
binary_accuracy: 0.9946 - val_loss: 0.1074 - val_binary_accuracy: 0.8683
Epoch 12/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0056 -
binary_accuracy: 0.9947 - val_loss: 0.1085 - val_binary_accuracy: 0.8674
Epoch 13/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0053 -
binary_accuracy: 0.9950 - val_loss: 0.1092 - val_binary_accuracy: 0.8686
Epoch 14/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0052 -
binary_accuracy: 0.9951 - val_loss: 0.1095 - val_binary_accuracy: 0.8697
Epoch 15/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0051 -
binary_accuracy: 0.9951 - val_loss: 0.1101 - val_binary_accuracy: 0.8687
Epoch 16/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0050 -
binary_accuracy: 0.9951 - val_loss: 0.1103 - val_binary_accuracy: 0.8695
Epoch 17/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0050 -
binary_accuracy: 0.9951 - val_loss: 0.1107 - val_binary_accuracy: 0.8688
Epoch 18/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0050 -
binary_accuracy: 0.9951 - val_loss: 0.1112 - val_binary_accuracy: 0.8686
```

```
Epoch 19/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0050 -
binary_accuracy: 0.9951 - val_loss: 0.1115 - val_binary_accuracy: 0.8682
Epoch 20/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0049 -
binary_accuracy: 0.9952 - val_loss: 0.1121 - val_binary_accuracy: 0.8677
```





```
[47]: 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 8ms/step - loss: 0.1271 - accuracy:
0.8371
Epoch 2/2
49/49 [==============================] - 0s 7ms/step - loss: 0.0581 - accuracy:
0.9275
782/782 [==============================] - 22s 28ms/step - loss: 0.0862 -
```

```
accuracy: 0.8832
```

[47]:

```
[0.0862470343708992, 0.8832399845123291]
```

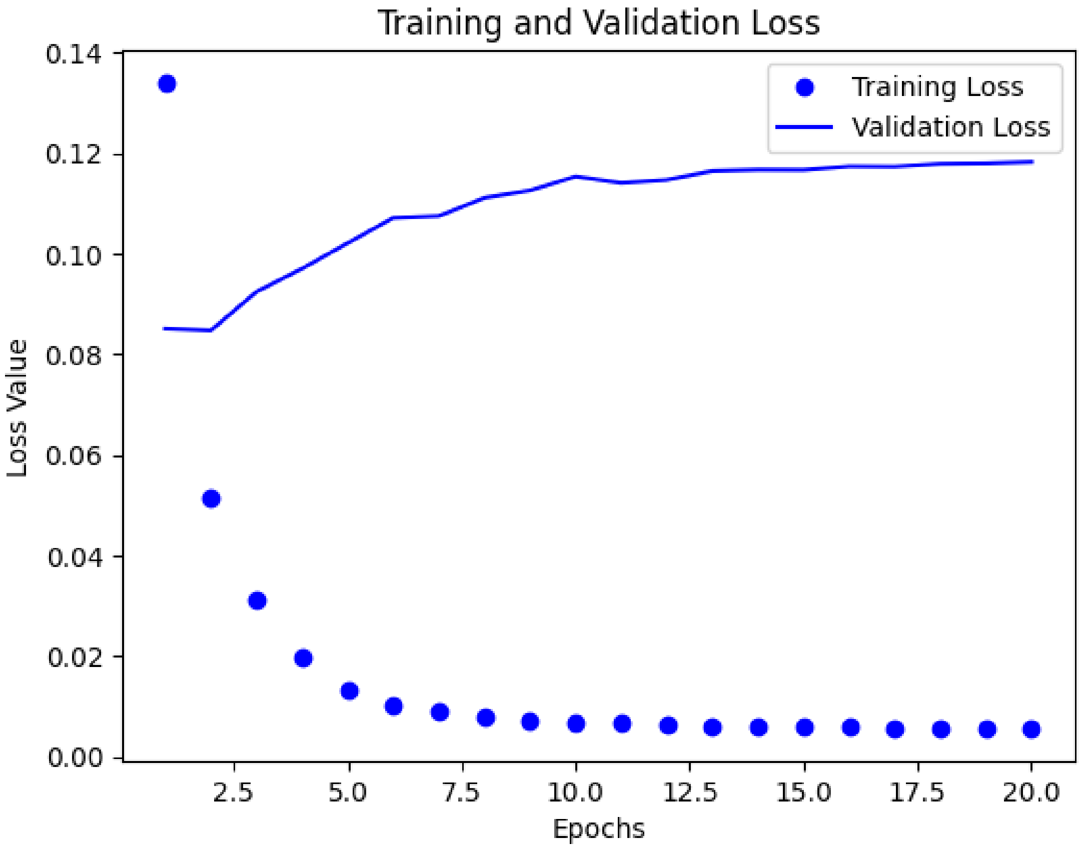
[48]:

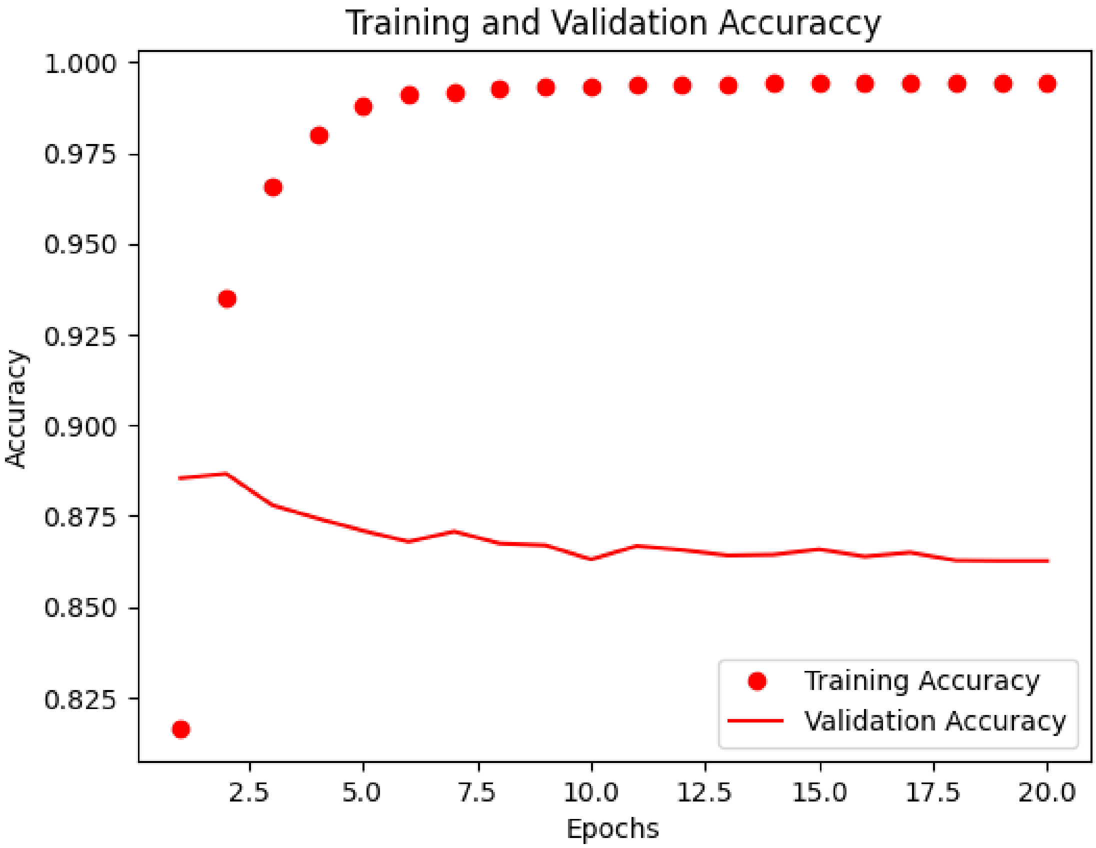
```python
#implemented two hideden layer with 64 neurons and mse loss function



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
```

```python
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 26ms/step - loss: 0.1340 -
binary_accuracy: 0.8165 - val_loss: 0.0851 - val_binary_accuracy: 0.8854
Epoch 2/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0515 -
binary_accuracy: 0.9349 - val_loss: 0.0847 - val_binary_accuracy: 0.8866
Epoch 3/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0313 -
binary_accuracy: 0.9656 - val_loss: 0.0924 - val_binary_accuracy: 0.8780
Epoch 4/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0198 -
binary_accuracy: 0.9802 - val_loss: 0.0970 - val_binary_accuracy: 0.8743
Epoch 5/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0134 -
```

```
binary_accuracy: 0.9883 - val_loss: 0.1021 - val_binary_accuracy: 0.8709
Epoch 6/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0102 -
binary_accuracy: 0.9913 - val_loss: 0.1071 - val_binary_accuracy: 0.8679
Epoch 7/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0092 -
binary_accuracy: 0.9918 - val_loss: 0.1075 - val_binary_accuracy: 0.8707
Epoch 8/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0080 -
binary_accuracy: 0.9927 - val_loss: 0.1111 - val_binary_accuracy: 0.8674
Epoch 9/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0072 -
binary_accuracy: 0.9933 - val_loss: 0.1126 - val_binary_accuracy: 0.8669
Epoch 10/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0067 -
binary_accuracy: 0.9935 - val_loss: 0.1153 - val_binary_accuracy: 0.8630
Epoch 11/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0067 -
binary_accuracy: 0.9936 - val_loss: 0.1141 - val_binary_accuracy: 0.8667
Epoch 12/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0064 -
binary_accuracy: 0.9937 - val_loss: 0.1147 - val_binary_accuracy: 0.8656
Epoch 13/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0061 -
binary_accuracy: 0.9941 - val_loss: 0.1164 - val_binary_accuracy: 0.8641
Epoch 14/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0059 -
binary_accuracy: 0.9941 - val_loss: 0.1167 - val_binary_accuracy: 0.8643
Epoch 15/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0058 -
binary_accuracy: 0.9941 - val_loss: 0.1166 - val_binary_accuracy: 0.8658
Epoch 16/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0058 -
binary_accuracy: 0.9943 - val_loss: 0.1173 - val_binary_accuracy: 0.8638
Epoch 17/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0057 -
binary_accuracy: 0.9943 - val_loss: 0.1173 - val_binary_accuracy: 0.8649
Epoch 18/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0057 -
binary_accuracy: 0.9944 - val_loss: 0.1179 - val_binary_accuracy: 0.8627
Epoch 19/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0056 -
binary_accuracy: 0.9944 - val_loss: 0.1180 - val_binary_accuracy: 0.8626
Epoch 20/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0056 -
binary_accuracy: 0.9944 - val_loss: 0.1182 - val_binary_accuracy: 0.8626
```


Training and Validation Loss
● Training Loss
Validation Loss
Loss Value
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epochs



[49]:
```
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 [==============================] - 1s 10ms/step - loss: 0.1112 - accuracy:
0.8504
Epoch 2/2
49/49 [==============================] - 0s 10ms/step - loss: 0.0527 - accuracy:
0.9329
782/782 [==============================] - 22s 28ms/step - loss: 0.0928 -
```

```
accuracy: 0.8758
```

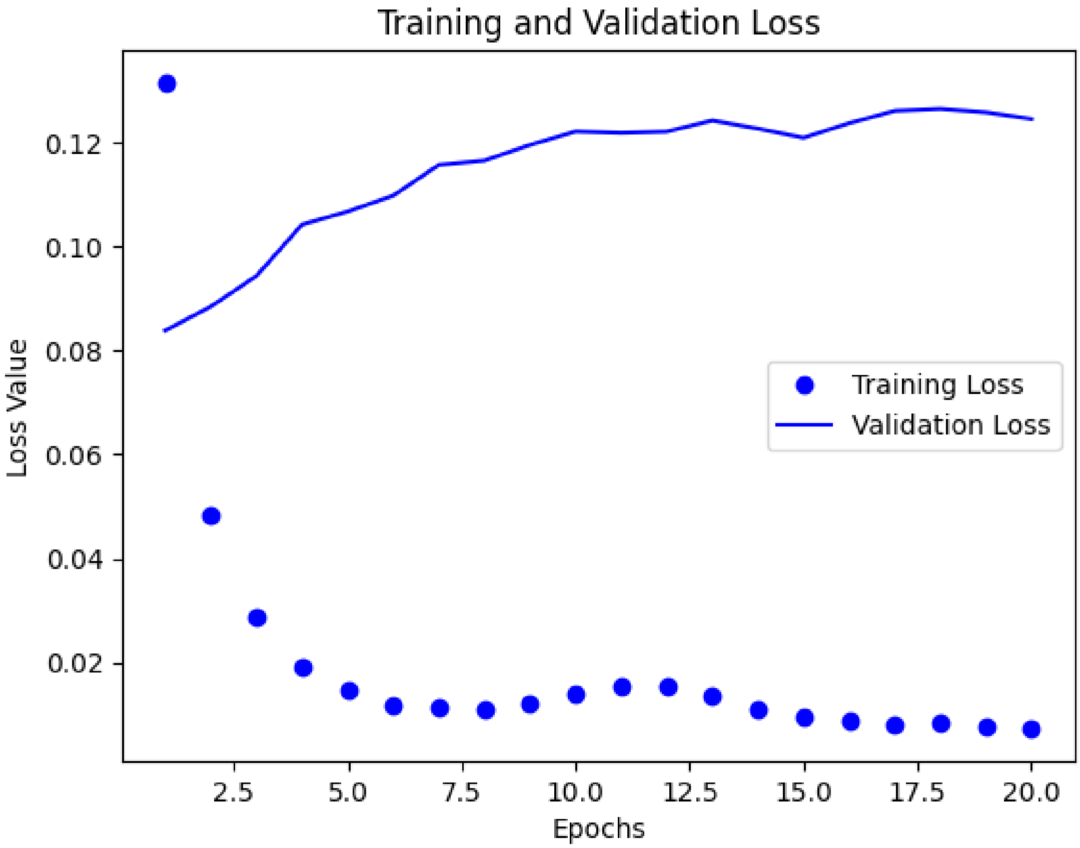
[49]:
```
[0.09279811382293701, 0.8758400082588196]
```

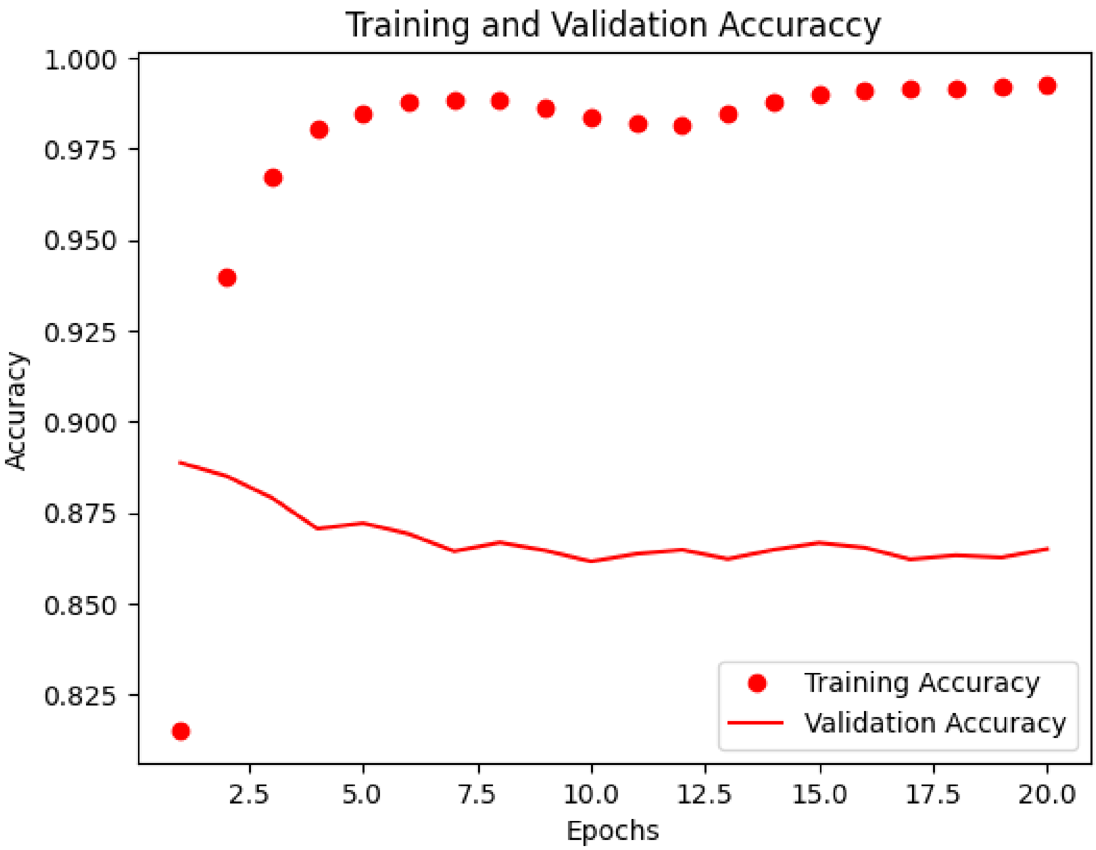
[50]:
```
#implemented two hideden layer with 128 neurons and mse loss function



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
```

```python
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: 0.1315 -
binary_accuracy: 0.8152 - val_loss: 0.0838 - val_binary_accuracy: 0.8887
Epoch 2/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0485 -
binary_accuracy: 0.9396 - val_loss: 0.0885 - val_binary_accuracy: 0.8851
Epoch 3/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0289 -
binary_accuracy: 0.9675 - val_loss: 0.0943 - val_binary_accuracy: 0.8791
Epoch 4/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0193 -
binary_accuracy: 0.9804 - val_loss: 0.1042 - val_binary_accuracy: 0.8706
Epoch 5/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0148 -
```

```
binary_accuracy: 0.9847 - val_loss: 0.1067 - val_binary_accuracy: 0.8721
Epoch 6/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0118 -
binary_accuracy: 0.9881 - val_loss: 0.1098 - val_binary_accuracy: 0.8692
Epoch 7/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0114 -
binary_accuracy: 0.9886 - val_loss: 0.1157 - val_binary_accuracy: 0.8644
Epoch 8/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0111 -
binary_accuracy: 0.9885 - val_loss: 0.1165 - val_binary_accuracy: 0.8668
Epoch 9/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0122 -
binary_accuracy: 0.9864 - val_loss: 0.1195 - val_binary_accuracy: 0.8646
Epoch 10/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0140 -
binary_accuracy: 0.9836 - val_loss: 0.1221 - val_binary_accuracy: 0.8616
Epoch 11/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0153 -
binary_accuracy: 0.9819 - val_loss: 0.1218 - val_binary_accuracy: 0.8637
Epoch 12/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0155 -
binary_accuracy: 0.9815 - val_loss: 0.1221 - val_binary_accuracy: 0.8648
Epoch 13/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0135 -
binary_accuracy: 0.9849 - val_loss: 0.1242 - val_binary_accuracy: 0.8623
Epoch 14/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0110 -
binary_accuracy: 0.9879 - val_loss: 0.1226 - val_binary_accuracy: 0.8648
Epoch 15/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0095 -
binary_accuracy: 0.9901 - val_loss: 0.1209 - val_binary_accuracy: 0.8667
Epoch 16/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0087 -
binary_accuracy: 0.9910 - val_loss: 0.1237 - val_binary_accuracy: 0.8654
Epoch 17/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0080 -
binary_accuracy: 0.9918 - val_loss: 0.1261 - val_binary_accuracy: 0.8622
Epoch 18/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0086 -
binary_accuracy: 0.9914 - val_loss: 0.1264 - val_binary_accuracy: 0.8633
Epoch 19/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0077 -
binary_accuracy: 0.9923 - val_loss: 0.1258 - val_binary_accuracy: 0.8627
Epoch 20/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0074 -
binary_accuracy: 0.9927 - val_loss: 0.1245 - val_binary_accuracy: 0.8650
```


Training and Validation Loss
Loss Value
0.12
0.10
0.08
0.06
0.04
0.02
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epochs
● Training Loss
Validation Loss



```
[64]: 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=1, batch_size=512)
      results = model.evaluate(x_test, y_test)

      results
```

```
49/49 [==============================] - 2s 17ms/step - loss: 0.1089 - accuracy:
0.8467
782/782 [==============================] - 4s 5ms/step - loss: 0.0905 -
accuracy: 0.8777
```

```
[64]: [0.09053931385278702, 0.8776800036430359]
```

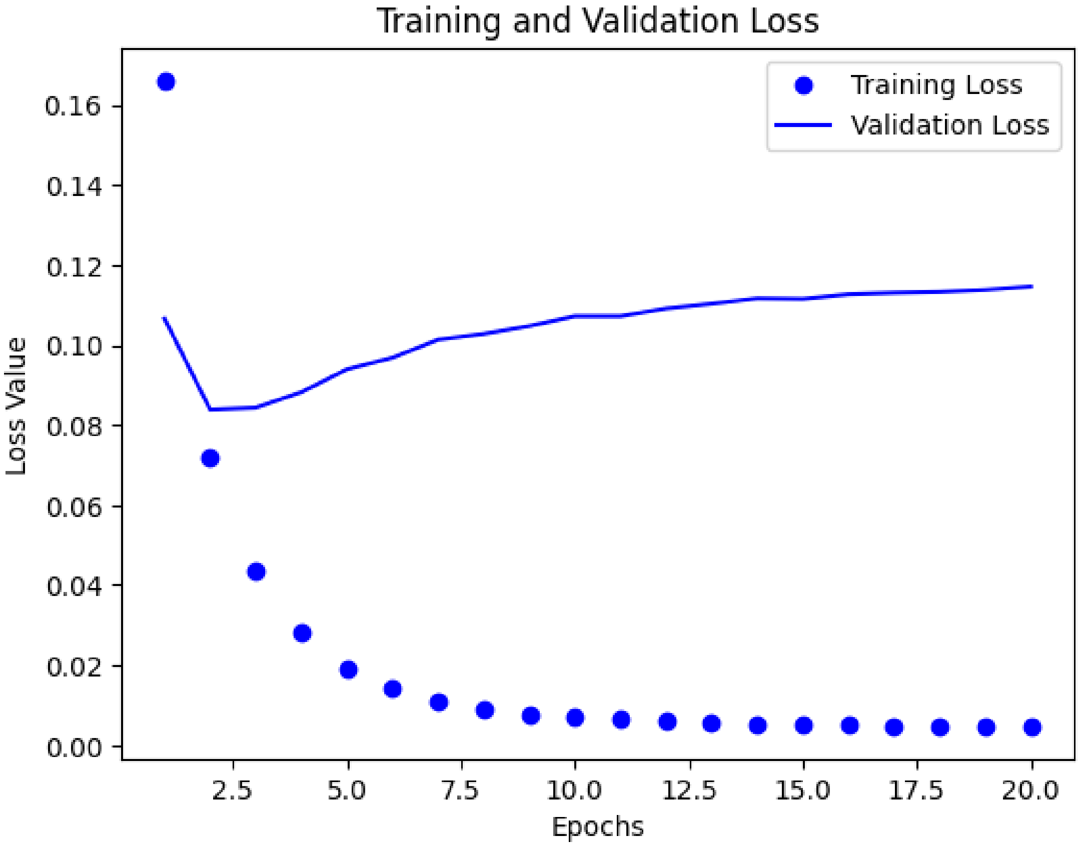
[52]:

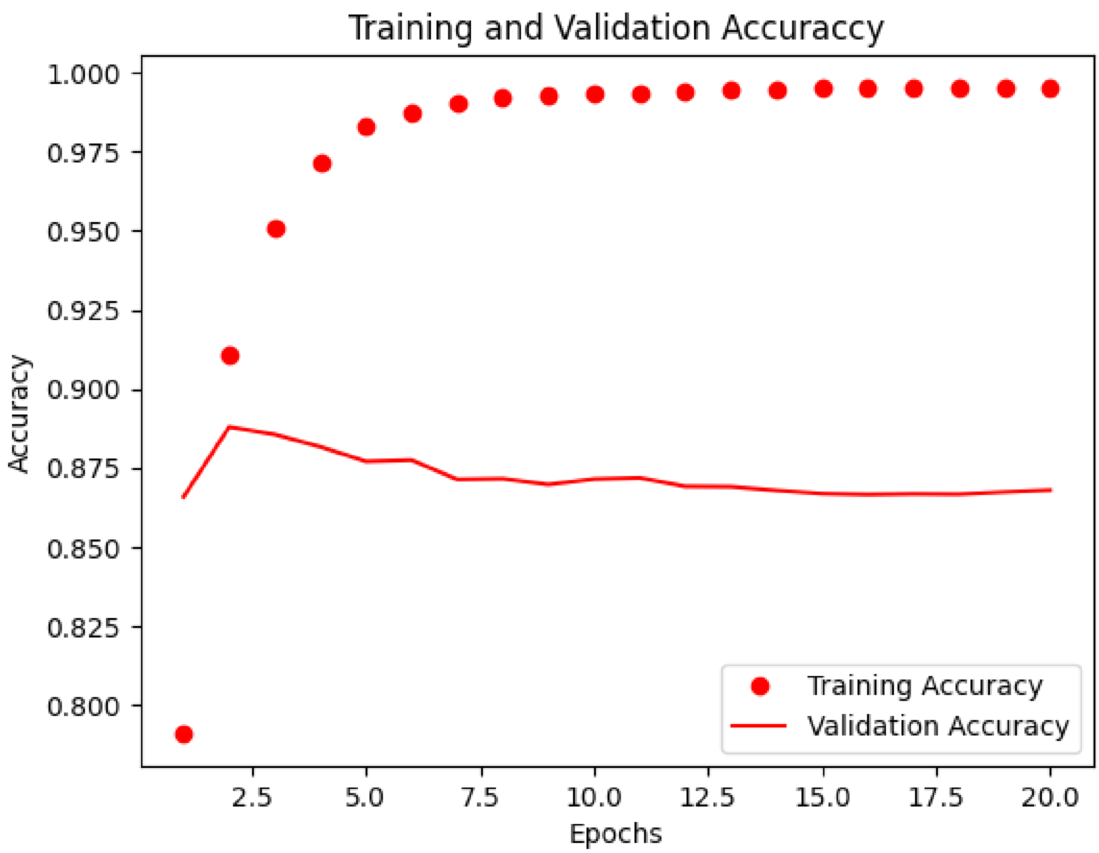
```
#implemented three hideden layer with 16 neurons and mse loss function



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 48ms/step - loss: 0.1661 -
binary_accuracy: 0.7909 - val_loss: 0.1066 - val_binary_accuracy: 0.8658
Epoch 2/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0721 -
binary_accuracy: 0.9108 - val_loss: 0.0839 - val_binary_accuracy: 0.8879
Epoch 3/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0434 -
binary_accuracy: 0.9511 - val_loss: 0.0844 - val_binary_accuracy: 0.8855
Epoch 4/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0281 -
binary_accuracy: 0.9717 - val_loss: 0.0882 - val_binary_accuracy: 0.8816
Epoch 5/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0192 -
binary_accuracy: 0.9829 - val_loss: 0.0939 - val_binary_accuracy: 0.8770
Epoch 6/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0143 -
binary_accuracy: 0.9874 - val_loss: 0.0968 - val_binary_accuracy: 0.8774
```

```
Epoch 7/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0109 -
binary_accuracy: 0.9904 - val_loss: 0.1013 - val_binary_accuracy: 0.8713
Epoch 8/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0090 -
binary_accuracy: 0.9920 - val_loss: 0.1028 - val_binary_accuracy: 0.8715
Epoch 9/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0078 -
binary_accuracy: 0.9929 - val_loss: 0.1047 - val_binary_accuracy: 0.8697
Epoch 10/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0070 -
binary_accuracy: 0.9935 - val_loss: 0.1072 - val_binary_accuracy: 0.8714
Epoch 11/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0066 -
binary_accuracy: 0.9937 - val_loss: 0.1072 - val_binary_accuracy: 0.8718
Epoch 12/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0063 -
binary_accuracy: 0.9941 - val_loss: 0.1091 - val_binary_accuracy: 0.8691
Epoch 13/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0058 -
binary_accuracy: 0.9945 - val_loss: 0.1103 - val_binary_accuracy: 0.8690
Epoch 14/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0054 -
binary_accuracy: 0.9947 - val_loss: 0.1116 - val_binary_accuracy: 0.8678
Epoch 15/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0052 -
binary_accuracy: 0.9950 - val_loss: 0.1115 - val_binary_accuracy: 0.8668
Epoch 16/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0050 -
binary_accuracy: 0.9951 - val_loss: 0.1127 - val_binary_accuracy: 0.8665
Epoch 17/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0049 -
binary_accuracy: 0.9952 - val_loss: 0.1130 - val_binary_accuracy: 0.8667
Epoch 18/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0049 -
binary_accuracy: 0.9952 - val_loss: 0.1133 - val_binary_accuracy: 0.8666
Epoch 19/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0048 -
binary_accuracy: 0.9953 - val_loss: 0.1138 - val_binary_accuracy: 0.8673
Epoch 20/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0048 -
binary_accuracy: 0.9953 - val_loss: 0.1146 - val_binary_accuracy: 0.8679
```


Training and Validation Loss
● Training Loss
Validation Loss
Loss Value
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
Epochs
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0



[65]:

```
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 8ms/step - loss: 0.1380 - accuracy:
0.8265
Epoch 2/2
49/49 [==============================] - 0s 7ms/step - loss: 0.0614 - accuracy:
0.9223
```

```
782/782 [==============================] - 25s 31ms/step - loss: 0.0862 -
accuracy: 0.8836
```

[65]:

```
[0.08623457700014114, 0.8835999965667725]
```

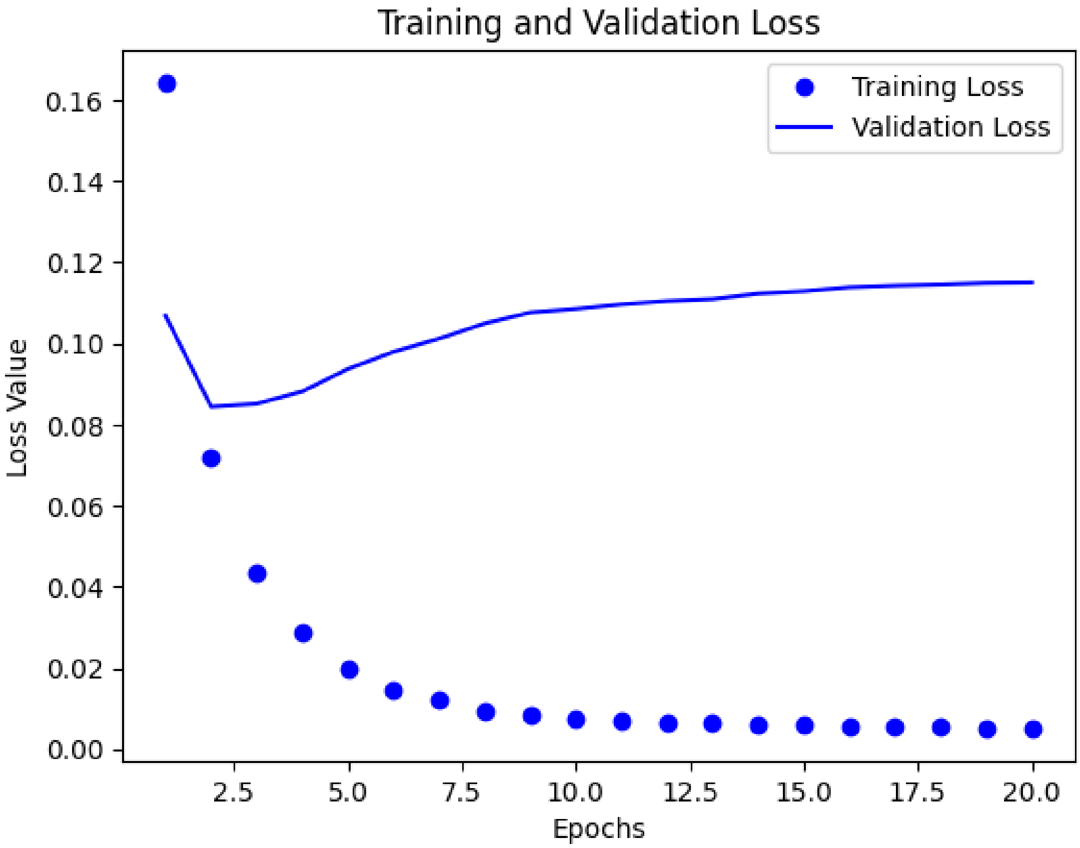
[54]:

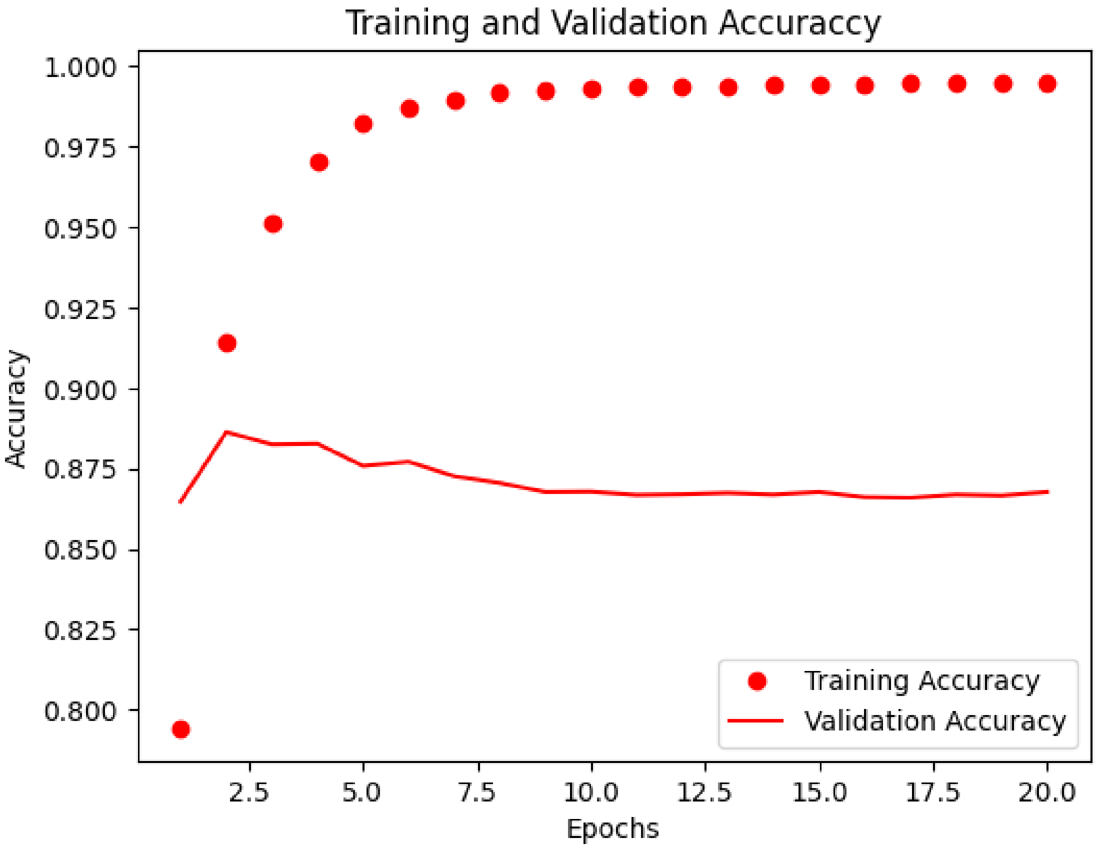
```python
    #implemented three hideden layer with 16 neurons and mse loss function and dropout



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
```

```python
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 25ms/step - loss: 0.1643 -
binary_accuracy: 0.7943 - val_loss: 0.1068 - val_binary_accuracy: 0.8647
Epoch 2/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0720 -
binary_accuracy: 0.9144 - val_loss: 0.0844 - val_binary_accuracy: 0.8863
Epoch 3/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0436 -
binary_accuracy: 0.9514 - val_loss: 0.0851 - val_binary_accuracy: 0.8825
Epoch 4/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0290 -
binary_accuracy: 0.9702 - val_loss: 0.0882 - val_binary_accuracy: 0.8827
```

```
Epoch 5/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0200 -
binary_accuracy: 0.9821 - val_loss: 0.0937 - val_binary_accuracy: 0.8758
Epoch 6/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0145 -
binary_accuracy: 0.9873 - val_loss: 0.0979 - val_binary_accuracy: 0.8771
Epoch 7/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0120 -
binary_accuracy: 0.9898 - val_loss: 0.1012 - val_binary_accuracy: 0.8726
Epoch 8/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0095 -
binary_accuracy: 0.9917 - val_loss: 0.1049 - val_binary_accuracy: 0.8705
Epoch 9/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0082 -
binary_accuracy: 0.9925 - val_loss: 0.1076 - val_binary_accuracy: 0.8677
Epoch 10/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0075 -
binary_accuracy: 0.9931 - val_loss: 0.1085 - val_binary_accuracy: 0.8678
Epoch 11/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0069 -
binary_accuracy: 0.9934 - val_loss: 0.1096 - val_binary_accuracy: 0.8668
Epoch 12/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0066 -
binary_accuracy: 0.9937 - val_loss: 0.1104 - val_binary_accuracy: 0.8670
Epoch 13/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0065 -
binary_accuracy: 0.9938 - val_loss: 0.1109 - val_binary_accuracy: 0.8674
Epoch 14/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0061 -
binary_accuracy: 0.9941 - val_loss: 0.1123 - val_binary_accuracy: 0.8669
Epoch 15/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0061 -
binary_accuracy: 0.9941 - val_loss: 0.1129 - val_binary_accuracy: 0.8677
Epoch 16/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0058 -
binary_accuracy: 0.9943 - val_loss: 0.1138 - val_binary_accuracy: 0.8661
Epoch 17/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0055 -
binary_accuracy: 0.9946 - val_loss: 0.1142 - val_binary_accuracy: 0.8659
Epoch 18/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0054 -
binary_accuracy: 0.9947 - val_loss: 0.1145 - val_binary_accuracy: 0.8669
Epoch 19/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0053 -
binary_accuracy: 0.9949 - val_loss: 0.1149 - val_binary_accuracy: 0.8666
Epoch 20/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0051 -
binary_accuracy: 0.9949 - val_loss: 0.1150 - val_binary_accuracy: 0.8677
```


Training and Validation Loss
● Training Loss
Validation Loss
Loss Value
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epochs



[66]:

```
model = models.Sequential()
model.add(layers.Dense(16, activation='tanh', input_shape=(10000,)))
layers.Dropout(0.5),
model.add(layers.Dense(16, activation='tanh', input_shape=(10000,)))
layers.Dropout(0.5),
model.add(layers.Dense(16, 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=2, batch_size=512)
results = model.evaluate(x_test, y_test)

results
```

```
Epoch 1/2
49/49 [==============================] - 2s 11ms/step - loss: 0.1356 - accuracy:
0.8301
```

```
Epoch 2/2
49/49 [==============================] - 0s 8ms/step - loss: 0.0604 - accuracy:
0.9228
782/782 [==============================] - 22s 28ms/step - loss: 0.0880 -
accuracy: 0.8816
```

[66]:
```
[0.08796077221632004, 0.8815600275993347]
```

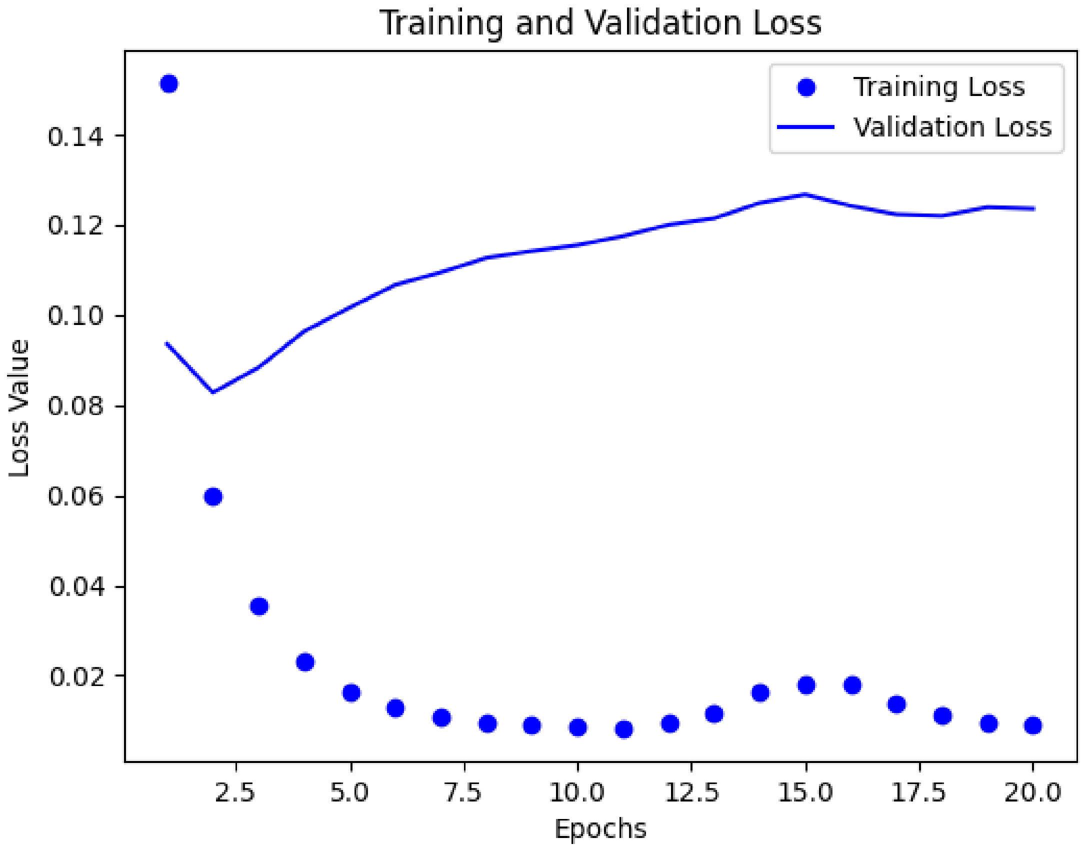
[56]:
```python
#implemented three hideden layer with 32 neurons and mse loss function



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))
```

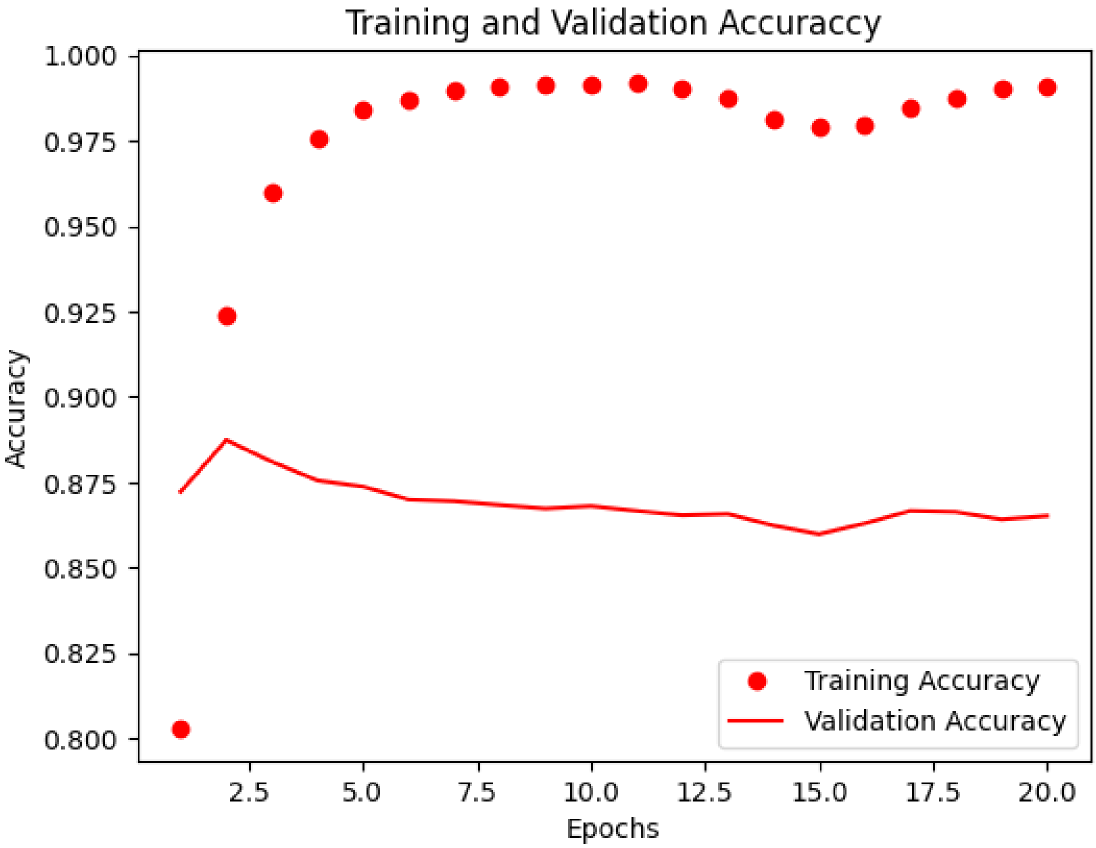
```
history_dict = history.history
history_dict.keys()
# Plotting the training and validation loss

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 [==============================] - 2s 38ms/step - loss: 0.1515 -
binary_accuracy: 0.8029 - val_loss: 0.0935 - val_binary_accuracy: 0.8722
Epoch 2/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0599 -
binary_accuracy: 0.9239 - val_loss: 0.0827 - val_binary_accuracy: 0.8873
Epoch 3/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0357 -
binary_accuracy: 0.9599 - val_loss: 0.0883 - val_binary_accuracy: 0.8810
Epoch 4/20
```

```
30/30 [==============================] - 0s 14ms/step - loss: 0.0233 -
binary_accuracy: 0.9754 - val_loss: 0.0963 - val_binary_accuracy: 0.8755
Epoch 5/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0163 -
binary_accuracy: 0.9841 - val_loss: 0.1016 - val_binary_accuracy: 0.8737
Epoch 6/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0128 -
binary_accuracy: 0.9869 - val_loss: 0.1066 - val_binary_accuracy: 0.8699
Epoch 7/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0109 -
binary_accuracy: 0.9897 - val_loss: 0.1094 - val_binary_accuracy: 0.8694
Epoch 8/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0095 -
binary_accuracy: 0.9911 - val_loss: 0.1126 - val_binary_accuracy: 0.8683
Epoch 9/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0090 -
binary_accuracy: 0.9913 - val_loss: 0.1141 - val_binary_accuracy: 0.8673
Epoch 10/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0087 -
binary_accuracy: 0.9916 - val_loss: 0.1154 - val_binary_accuracy: 0.8680
Epoch 11/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0083 -
binary_accuracy: 0.9920 - val_loss: 0.1174 - val_binary_accuracy: 0.8666
Epoch 12/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0094 -
binary_accuracy: 0.9905 - val_loss: 0.1199 - val_binary_accuracy: 0.8653
Epoch 13/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0116 -
binary_accuracy: 0.9873 - val_loss: 0.1214 - val_binary_accuracy: 0.8657
Epoch 14/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0162 -
binary_accuracy: 0.9814 - val_loss: 0.1248 - val_binary_accuracy: 0.8623
Epoch 15/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0181 -
binary_accuracy: 0.9788 - val_loss: 0.1266 - val_binary_accuracy: 0.8598
Epoch 16/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0179 -
binary_accuracy: 0.9795 - val_loss: 0.1241 - val_binary_accuracy: 0.8629
Epoch 17/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0137 -
binary_accuracy: 0.9845 - val_loss: 0.1222 - val_binary_accuracy: 0.8666
Epoch 18/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0112 -
binary_accuracy: 0.9877 - val_loss: 0.1219 - val_binary_accuracy: 0.8663
Epoch 19/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0094 -
binary_accuracy: 0.9905 - val_loss: 0.1238 - val_binary_accuracy: 0.8641
Epoch 20/20
```

```
30/30 [==============================] - 0s 13ms/step - loss: 0.0090 -
binary_accuracy: 0.9907 - val_loss: 0.1235 - val_binary_accuracy: 0.8651
```





[57]:

```
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 [==============================] - 1s 8ms/step - loss: 0.1198 - accuracy:
0.8392
Epoch 2/2
49/49 [==============================] - 0s 7ms/step - loss: 0.0542 - accuracy:
0.9314
```

```
782/782 [==============================] - 23s 29ms/step - loss: 0.0903 -
accuracy: 0.8784
```

[57]:

```
[0.09027967602014542, 0.8784000277519226]
```

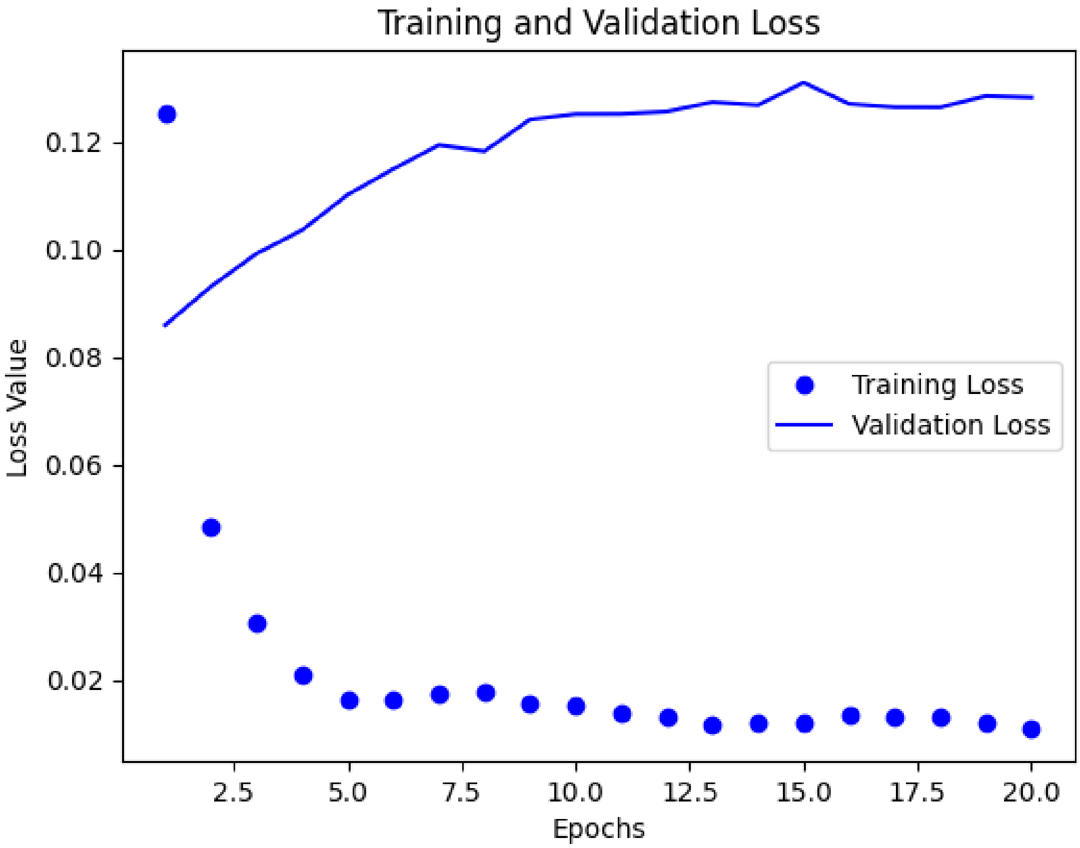
[58]:

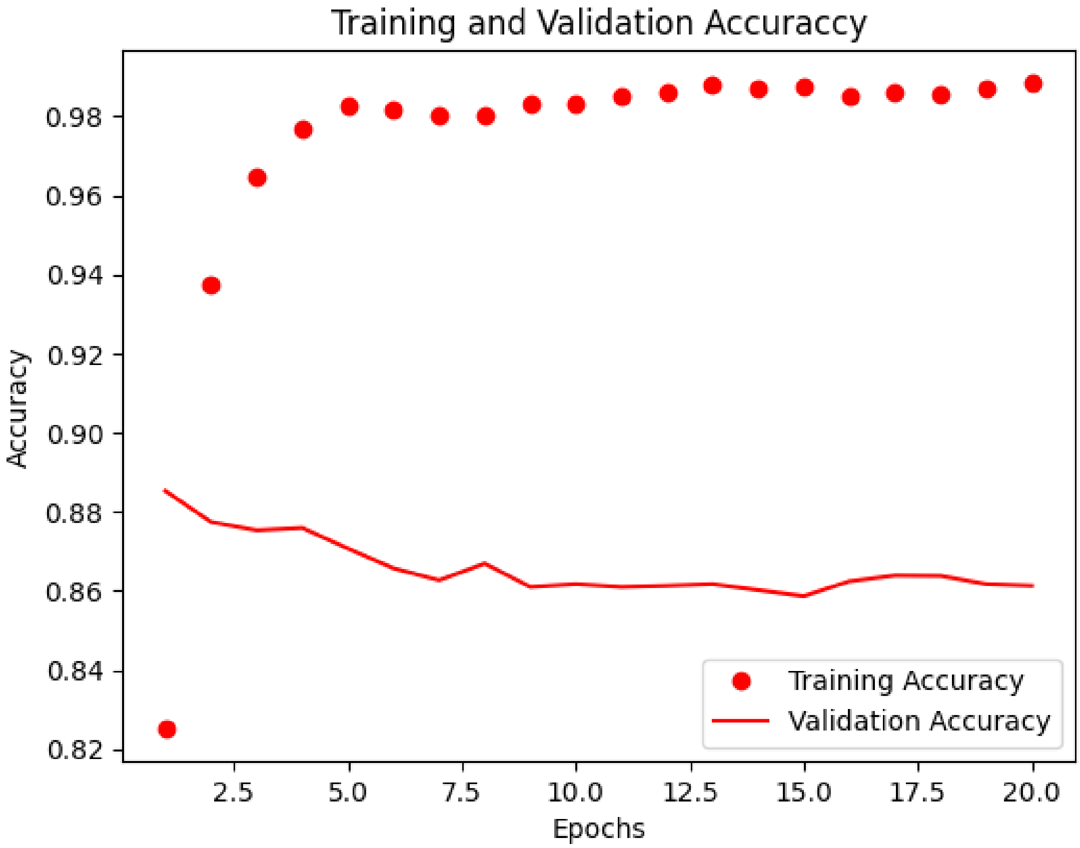
```python
#implemented three hideden layer with 64 neurons and mse loss function



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()
```

```python
# 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 27ms/step - loss: 0.1251 -
binary_accuracy: 0.8253 - val_loss: 0.0859 - val_binary_accuracy: 0.8852
Epoch 2/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0484 -
binary_accuracy: 0.9375 - val_loss: 0.0931 - val_binary_accuracy: 0.8774
Epoch 3/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0304 -
binary_accuracy: 0.9648 - val_loss: 0.0992 - val_binary_accuracy: 0.8753
Epoch 4/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0209 -
binary_accuracy: 0.9768 - val_loss: 0.1036 - val_binary_accuracy: 0.8759
Epoch 5/20
```

```
30/30 [==============================] - 0s 15ms/step - loss: 0.0162 -
binary_accuracy: 0.9828 - val_loss: 0.1101 - val_binary_accuracy: 0.8707
Epoch 6/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0162 -
binary_accuracy: 0.9815 - val_loss: 0.1149 - val_binary_accuracy: 0.8657
Epoch 7/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0175 -
binary_accuracy: 0.9801 - val_loss: 0.1194 - val_binary_accuracy: 0.8627
Epoch 8/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0175 -
binary_accuracy: 0.9803 - val_loss: 0.1182 - val_binary_accuracy: 0.8669
Epoch 9/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0156 -
binary_accuracy: 0.9829 - val_loss: 0.1241 - val_binary_accuracy: 0.8610
Epoch 10/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0151 -
binary_accuracy: 0.9833 - val_loss: 0.1251 - val_binary_accuracy: 0.8617
Epoch 11/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0138 -
binary_accuracy: 0.9851 - val_loss: 0.1252 - val_binary_accuracy: 0.8610
Epoch 12/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0130 -
binary_accuracy: 0.9861 - val_loss: 0.1256 - val_binary_accuracy: 0.8613
Epoch 13/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0116 -
binary_accuracy: 0.9878 - val_loss: 0.1273 - val_binary_accuracy: 0.8617
Epoch 14/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0120 -
binary_accuracy: 0.9871 - val_loss: 0.1268 - val_binary_accuracy: 0.8602
Epoch 15/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0120 -
binary_accuracy: 0.9873 - val_loss: 0.1310 - val_binary_accuracy: 0.8587
Epoch 16/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0135 -
binary_accuracy: 0.9851 - val_loss: 0.1270 - val_binary_accuracy: 0.8624
Epoch 17/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0131 -
binary_accuracy: 0.9859 - val_loss: 0.1264 - val_binary_accuracy: 0.8639
Epoch 18/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0129 -
binary_accuracy: 0.9857 - val_loss: 0.1264 - val_binary_accuracy: 0.8638
Epoch 19/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0121 -
binary_accuracy: 0.9867 - val_loss: 0.1285 - val_binary_accuracy: 0.8617
Epoch 20/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0110 -
binary_accuracy: 0.9885 - val_loss: 0.1282 - val_binary_accuracy: 0.8613
```


Training and Validation Loss
Loss Value
0.12
0.10
0.08
0.06
0.04
0.02
Epochs
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
● Training Loss
Validation Loss



[67]:

```
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=1, batch_size=512)
results = model.evaluate(x_test, y_test)

results
```

```
49/49 [==============================] - 2s 18ms/step - loss: 0.1117 - accuracy:
0.8441
782/782 [==============================] - 27s 34ms/step - loss: 0.0885 -
accuracy: 0.8803
```

[67]:

```
[0.08847083896398544, 0.8803200125694275]
```

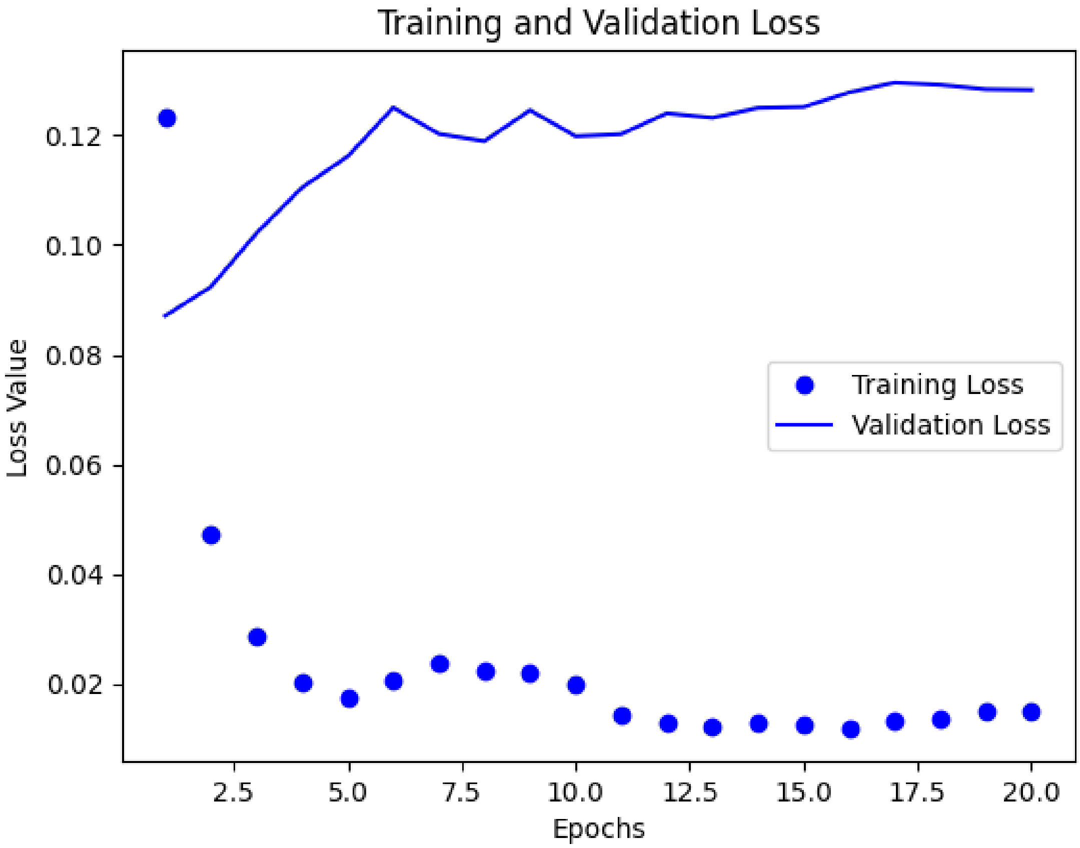
[60]:

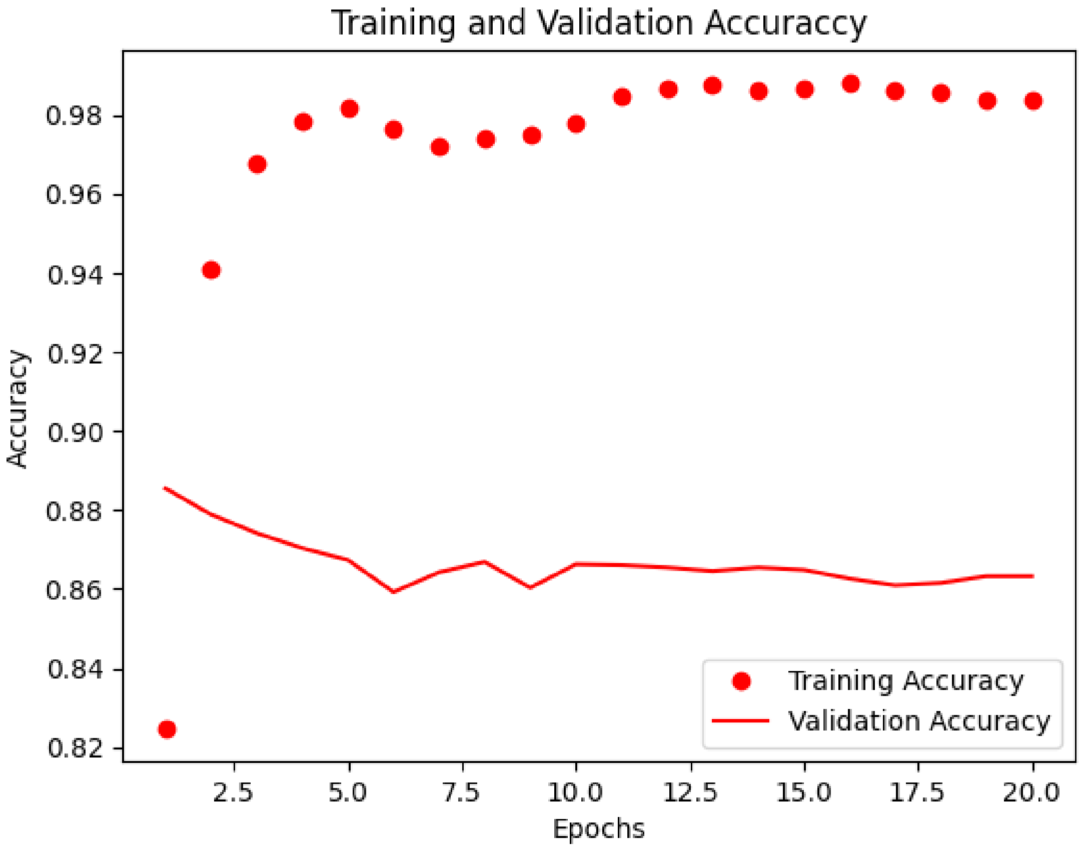
```
#implemented three hideden layer with 128 neurons and mse loss function


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 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: 0.1232 -
binary_accuracy: 0.8247 - val_loss: 0.0871 - val_binary_accuracy: 0.8854
Epoch 2/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0473 -
binary_accuracy: 0.9410 - val_loss: 0.0924 - val_binary_accuracy: 0.8788
Epoch 3/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0288 -
binary_accuracy: 0.9675 - val_loss: 0.1021 - val_binary_accuracy: 0.8741
Epoch 4/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0203 -
binary_accuracy: 0.9785 - val_loss: 0.1105 - val_binary_accuracy: 0.8703
Epoch 5/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0174 -
binary_accuracy: 0.9816 - val_loss: 0.1161 - val_binary_accuracy: 0.8673
Epoch 6/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0206 -
binary_accuracy: 0.9766 - val_loss: 0.1250 - val_binary_accuracy: 0.8592
```

```
Epoch 7/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0238 -
binary_accuracy: 0.9720 - val_loss: 0.1203 - val_binary_accuracy: 0.8642
Epoch 8/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0223 -
binary_accuracy: 0.9738 - val_loss: 0.1190 - val_binary_accuracy: 0.8668
Epoch 9/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0221 -
binary_accuracy: 0.9747 - val_loss: 0.1246 - val_binary_accuracy: 0.8603
Epoch 10/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0200 -
binary_accuracy: 0.9777 - val_loss: 0.1198 - val_binary_accuracy: 0.8662
Epoch 11/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0143 -
binary_accuracy: 0.9848 - val_loss: 0.1202 - val_binary_accuracy: 0.8660
Epoch 12/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0129 -
binary_accuracy: 0.9867 - val_loss: 0.1240 - val_binary_accuracy: 0.8654
Epoch 13/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0122 -
binary_accuracy: 0.9873 - val_loss: 0.1232 - val_binary_accuracy: 0.8645
Epoch 14/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0129 -
binary_accuracy: 0.9863 - val_loss: 0.1250 - val_binary_accuracy: 0.8654
Epoch 15/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0127 -
binary_accuracy: 0.9868 - val_loss: 0.1252 - val_binary_accuracy: 0.8648
Epoch 16/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0119 -
binary_accuracy: 0.9881 - val_loss: 0.1278 - val_binary_accuracy: 0.8626
Epoch 17/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0133 -
binary_accuracy: 0.9859 - val_loss: 0.1296 - val_binary_accuracy: 0.8609
Epoch 18/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0137 -
binary_accuracy: 0.9855 - val_loss: 0.1292 - val_binary_accuracy: 0.8615
Epoch 19/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0150 -
binary_accuracy: 0.9839 - val_loss: 0.1284 - val_binary_accuracy: 0.8632
Epoch 20/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0150 -
binary_accuracy: 0.9837 - val_loss: 0.1283 - val_binary_accuracy: 0.8632
```


Training and Validation Loss
0.12
0.10
0.08
0.06
0.04
0.02
Loss Value
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epochs
● Training Loss
— Validation Loss



[68]:
```
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=1, batch_size=512)
results = model.evaluate(x_test, y_test)

results
```

```
49/49 [==============================] - 2s 18ms/step - loss: 0.1065 - accuracy:
0.8508
782/782 [==============================] - 6s 7ms/step - loss: 0.0892 -
accuracy: 0.8785
```

[68]:
```
[0.08922901004552841, 0.8785200119018555]
```

```
[70]: pip install nbconvert
```

```
Requirement already satisfied: nbconvert in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (6.5.4)
Requirement already satisfied: tinycss2 in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from nbconvert) (1.2.1)
Requirement already satisfied: beautifulsoup4 in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from nbconvert) (4.11.1)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from nbconvert) (2.1.1)
Requirement already satisfied: bleach in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from nbconvert) (4.1.0)
Requirement already satisfied: mistune<2,>=0.8.1 in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from nbconvert) (0.8.4)
Requirement already satisfied: packaging in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from nbconvert) (22.0)
Requirement already satisfied: lxml in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from nbconvert) (4.9.1)
Requirement already satisfied: pygments>=2.4.1 in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from nbconvert) (2.11.2)
Requirement already satisfied: defusedxml in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from nbconvert) (0.7.1)
Requirement already satisfied: entrypoints>=0.2.2 in
c:\users\akhil\anaconda3\new folder\envs\tensorflow\lib\site-packages (from
nbconvert) (0.4)
Requirement already satisfied: jinja2>=3.0 in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from nbconvert) (3.1.2)
Requirement already satisfied: jupyter-core>=4.7 in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from nbconvert) (5.2.0)
Requirement already satisfied: jupyterlab-pygments in
c:\users\akhil\anaconda3\new folder\envs\tensorflow\lib\site-packages (from
nbconvert) (0.1.2)
Requirement already satisfied: traitlets>=5.0 in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from nbconvert) (5.7.1)
Requirement already satisfied: nbformat>=5.1 in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from nbconvert) (5.7.0)
Requirement already satisfied: nbclient>=0.5.0 in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from nbconvert) (0.5.13)
Requirement already satisfied: pandocfilters>=1.4.1 in
c:\users\akhil\anaconda3\new folder\envs\tensorflow\lib\site-packages (from
nbconvert) (1.5.0)
Requirement already satisfied: pywin32>=1.0 in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from jupyter-core>=4.7->nbconvert)
(305.1)
Requirement already satisfied: platformdirs>=2.5 in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from jupyter-core>=4.7->nbconvert)
(2.5.2)
Requirement already satisfied: jupyter-client>=6.1.5 in
```

```
c:\users\akhil\anaconda3\new folder\envs\tensorflow\lib\site-packages (from
nbclient>=0.5.0->nbconvert) (7.4.9)
Requirement already satisfied: nest-asyncio in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from nbclient>=0.5.0->nbconvert)
(1.5.6)
Requirement already satisfied: fastjsonschema in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from nbformat>=5.1->nbconvert)
(2.16.2)
Requirement already satisfied: jsonschema>=2.6 in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from nbformat>=5.1->nbconvert)
(4.17.3)
Requirement already satisfied: soupsieve>1.2 in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from beautifulsoup4->nbconvert)
(2.3.2.post1)
Requirement already satisfied: webencodings in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from bleach->nbconvert) (0.5.1)
Requirement already satisfied: six>=1.9.0 in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from bleach->nbconvert) (1.16.0)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in
c:\users\akhil\anaconda3\new folder\envs\tensorflow\lib\site-packages (from
jsonschema>=2.6->nbformat>=5.1->nbconvert) (0.18.0)
Requirement already satisfied: attrs>=17.4.0 in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from
jsonschema>=2.6->nbformat>=5.1->nbconvert) (22.1.0)
Requirement already satisfied: pyzmq>=23.0 in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from jupyter-
client>=6.1.5->nbclient>=0.5.0->nbconvert) (23.2.0)
Requirement already satisfied: tornado>=6.2 in c:\users\akhil\anaconda3\new
folder\envs\tensorflow\lib\site-packages (from jupyter-
client>=6.1.5->nbclient>=0.5.0->nbconvert) (6.2)
Requirement already satisfied: python-dateutil>=2.8.2 in
c:\users\akhil\anaconda3\new folder\envs\tensorflow\lib\site-packages (from
jupyter-client>=6.1.5->nbclient>=0.5.0->nbconvert) (2.8.2)
Note: you may need to restart the kernel to use updated packages.
```

[ ]: