In [ ]:

```
**Assignment-1 : Neural Networks**
Name : Vijay Charan Reddy Gottam
```

In [1]:

```
!pip install tensorflow
```

```
Requirement already satisfied: tensorflow in c:\users\vijay\anaconda3\lib\site-pac
kages (2.15.0)
Requirement already satisfied: tensorflow-intel==2.15.0 in c:\users\vijay\anaconda
3\lib\site-packages (from tensorflow) (2.15.0)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\vijay\anaconda3\lib\site
-packages (from tensorflow-intel==2.15.0->tensorflow) (2.1.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\vijay\anaconda3\lib\s
ite-packages (from tensorflow-intel==2.15.0->tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in c:\users\vijay\anaconda3\li
b\site-packages (from tensorflow-intel==2.15.0->tensorflow) (23.5.26)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in c:\users\vij
ay\anaconda3\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow) (0.5.4)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\vijay\anaconda3\lib
\site-packages (from tensorflow-intel==2.15.0->tensorflow) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in c:\users\vijay\anaconda3\lib\site-pa
ckages (from tensorflow-intel==2.15.0->tensorflow) (3.9.0)
Requirement already satisfied: libclang>=13.0.0 in c:\users\vijay\anaconda3\lib\si
te-packages (from tensorflow-intel==2.15.0->tensorflow) (16.0.6)
Requirement already satisfied: ml-dtypes~=0.2.0 in c:\users\vijay\anaconda3\lib\si
te-packages (from tensorflow-intel==2.15.0->tensorflow) (0.2.0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in c:\users\vijay\anaconda3\li
b\site-packages (from tensorflow-intel==2.15.0->tensorflow) (1.24.3)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\vijay\anaconda3\lib\s
ite-packages (from tensorflow-intel==2.15.0->tensorflow) (3.3.0)
Requirement already satisfied: packaging in c:\users\vijay\anaconda3\lib\site-pack
ages (from tensorflow-intel==2.15.0->tensorflow) (23.1)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.
4,!=4.21.5,<5.0.0dev,>=3.20.3 in c:\users\vijay\anaconda3\lib\site-packages (from
tensorflow-intel==2.15.0->tensorflow) (4.25.3)
Requirement already satisfied: setuptools in c:\users\vijay\anaconda3\lib\site-pac
kages (from tensorflow-intel==2.15.0->tensorflow) (68.0.0)
Requirement already satisfied: six>=1.12.0 in c:\users\vijay\anaconda3\lib\site-pa
ckages (from tensorflow-intel==2.15.0->tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\vijay\anaconda3\lib\si
te-packages (from tensorflow-intel==2.15.0->tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\vijay\anaconda
3\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow) (4.7.1)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in c:\users\vijay\anaconda3\lib
\site-packages (from tensorflow-intel==2.15.0->tensorflow) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in c:\users\vi
jay\anaconda3\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow) (0.31.
0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\vijay\anaconda3\lib
\site-packages (from tensorflow-intel==2.15.0->tensorflow) (1.62.0)
Requirement already satisfied: tensorboard<2.16,>=2.15 in c:\users\vijay\anaconda3
\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow) (2.15.2)
Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0 in c:\users\vija
y\anaconda3\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow) (2.15.0)
Requirement already satisfied: keras<2.16,>=2.15.0 in c:\users\vijay\anaconda3\lib
\site-packages (from tensorflow-intel==2.15.0->tensorflow) (2.15.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\vijay\anaconda3\lib
\site-packages (from astunparse>=1.6.0->tensorflow-intel==2.15.0->tensorflow) (0.3
8.4)
Requirement already satisfied: google-auth<3,>=1.6.3 in c:\users\vijay\anaconda3\l
ib\site-packages (from tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorfl
ow) (2.28.1)
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in c:\users\vijay\anac
onda3\lib\site-packages (from tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->t
ensorflow) (1.2.0)
Requirement already satisfied: markdown>=2.6.8 in c:\users\vijay\anaconda3\lib\sit
e-packages (from tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow)
(3.4.1)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\vijay\anaconda3\lib
\site-packages (from tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflo
w) (2.31.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in c:\users\v
ijay\anaconda3\lib\site-packages (from tensorboard<2.16,>=2.15->tensorflow-intel==
2.15.0->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\vijay\anaconda3\lib\sit
e-packages (from tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow)
(2.2.3)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in c:\users\vijay\anaconda3
\lib\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflo
w-intel==2.15.0->tensorflow) (5.3.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\users\vijay\anaconda3\l
ib\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow-
intel==2.15.0->tensorflow) (0.2.8)
Requirement already satisfied: rsa<5,>=3.1.4 in c:\users\vijay\anaconda3\lib\site-
packages (from google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow-intel==
2.15.0->tensorflow) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in c:\users\vijay\anaconda
3\lib\site-packages (from google-auth-oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->t
ensorflow-intel==2.15.0->tensorflow) (1.3.1)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\vijay\anaconda
3\lib\site-packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow
-intel==2.15.0->tensorflow) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\vijay\anaconda3\lib\site-p
ackages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow-intel==2.1
5.0->tensorflow) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\vijay\anaconda3\lib
\site-packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow-inte
l==2.15.0->tensorflow) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\vijay\anaconda3\lib
\site-packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow-inte
l==2.15.0->tensorflow) (2023.7.22)
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\users\vijay\anaconda3\lib\s
ite-packages (from werkzeug>=1.0.1->tensorboard<2.16,>=2.15->tensorflow-intel==2.1
5.0->tensorflow) (2.1.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in c:\users\vijay\anaconda3\li
b\site-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.
16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in c:\users\vijay\anaconda3\lib\sit
e-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<2,>=0.5->tensorboa
rd<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (3.2.2)
```

In [2]:

```
#Install Tensorflow
```

In [3]:

```
#The "IMDB dataset," a compilation of 50,000 sharply divided reviews from the Inter
```

In [4]:

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

```
WARNING:tensorflow:From C:\Users\Vijay\anaconda3\Lib\site-packages\keras\src\losse
s.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please u
se tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

In [5]:

```
print(train_data,train_data.shape)
```

```
[list([1, 14, 22, 16, 43, 530, 973, 1622, 1385, 65, 458, 4468, 66, 3941, 4, 173, 3
6, 256, 5, 25, 100, 43, 838, 112, 50, 670, 2, 9, 35, 480, 284, 5, 150, 4, 172, 11
2, 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, 1
24, 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, 1
41, 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])
 list([1, 194, 1153, 194, 8255, 78, 228, 5, 6, 1463, 4369, 5012, 134, 26, 4, 715,
8, 118, 1634, 14, 394, 20, 13, 119, 954, 189, 102, 5, 207, 110, 3103, 21, 14, 69,
188, 8, 30, 23, 7, 4, 249, 126, 93, 4, 114, 9, 2300, 1523, 5, 647, 4, 116, 9, 35,
8163, 4, 229, 9, 340, 1322, 4, 118, 9, 4, 130, 4901, 19, 4, 1002, 5, 89, 29, 952,
46, 37, 4, 455, 9, 45, 43, 38, 1543, 1905, 398, 4, 1649, 26, 6853, 5, 163, 11, 321
5, 2, 4, 1153, 9, 194, 775, 7, 8255, 2, 349, 2637, 148, 605, 2, 8003, 15, 123, 12
5, 68, 2, 6853, 15, 349, 165, 4362, 98, 5, 4, 228, 9, 43, 2, 1157, 15, 299, 120,
5, 120, 174, 11, 220, 175, 136, 50, 9, 4373, 228, 8255, 5, 2, 656, 245, 2350, 5,
4, 9837, 131, 152, 491, 18, 2, 32, 7464, 1212, 14, 9, 6, 371, 78, 22, 625, 64, 138
2, 9, 8, 168, 145, 23, 4, 1690, 15, 16, 4, 1355, 5, 28, 6, 52, 154, 462, 33, 89, 7
8, 285, 16, 145, 95])
 list([1, 14, 47, 8, 30, 31, 7, 4, 249, 108, 7, 4, 5974, 54, 61, 369, 13, 71, 149,
14, 22, 112, 4, 2401, 311, 12, 16, 3711, 33, 75, 43, 1829, 296, 4, 86, 320, 35, 53
4, 19, 263, 4821, 1301, 4, 1873, 33, 89, 78, 12, 66, 16, 4, 360, 7, 4, 58, 316, 33
4, 11, 4, 1716, 43, 645, 662, 8, 257, 85, 1200, 42, 1228, 2578, 83, 68, 3912, 15,
36, 165, 1539, 278, 36, 69, 2, 780, 8, 106, 14, 6905, 1338, 18, 6, 22, 12, 215, 2
8, 610, 40, 6, 87, 326, 23, 2300, 21, 23, 22, 12, 272, 40, 57, 31, 11, 4, 22, 47,
6, 2307, 51, 9, 170, 23, 595, 116, 595, 1352, 13, 191, 79, 638, 89, 2, 14, 9, 8, 1
06, 607, 624, 35, 534, 6, 227, 7, 129, 113])
 ...
 list([1, 11, 6, 230, 245, 6401, 9, 6, 1225, 446, 2, 45, 2174, 84, 8322, 4007, 21,
4, 912, 84, 2, 325, 725, 134, 2, 1715, 84, 5, 36, 28, 57, 1099, 21, 8, 140, 8, 70
3, 5, 2, 84, 56, 18, 1644, 14, 9, 31, 7, 4, 9406, 1209, 2295, 2, 1008, 18, 6, 20,
207, 110, 563, 12, 8, 2901, 2, 8, 97, 6, 20, 53, 4767, 74, 4, 460, 364, 1273, 29,
270, 11, 960, 108, 45, 40, 29, 2961, 395, 11, 6, 4065, 500, 7, 2, 89, 364, 70, 29,
140, 4, 64, 4780, 11, 4, 2678, 26, 178, 4, 529, 443, 2, 5, 27, 710, 117, 2, 8123,
165, 47, 84, 37, 131, 818, 14, 595, 10, 10, 61, 1242, 1209, 10, 10, 288, 2260, 170
2, 34, 2901, 2, 4, 65, 496, 4, 231, 7, 790, 5, 6, 320, 234, 2766, 234, 1119, 1574,
7, 496, 4, 139, 929, 2901, 2, 7750, 5, 4241, 18, 4, 8497, 2, 250, 11, 1818, 7561,
4, 4217, 5408, 747, 1115, 372, 1890, 1006, 541, 9303, 7, 4, 59, 2, 4, 3586, 2])
 list([1, 1446, 7079, 69, 72, 3305, 13, 610, 930, 8, 12, 582, 23, 5, 16, 484, 685,
54, 349, 11, 4120, 2959, 45, 58, 1466, 13, 197, 12, 16, 43, 23, 2, 5, 62, 30, 145,
402, 11, 4131, 51, 575, 32, 61, 369, 71, 66, 770, 12, 1054, 75, 100, 2198, 8, 4, 1
05, 37, 69, 147, 712, 75, 3543, 44, 257, 390, 5, 69, 263, 514, 105, 50, 286, 1814,
23, 4, 123, 13, 161, 40, 5, 421, 4, 116, 16, 897, 13, 2, 40, 319, 5872, 112, 6700,
11, 4803, 121, 25, 70, 3468, 4, 719, 3798, 13, 18, 31, 62, 40, 8, 7200, 4, 2, 7, 1
4, 123, 5, 942, 25, 8, 721, 12, 145, 5, 202, 12, 160, 580, 202, 12, 6, 52, 58, 2,
92, 401, 728, 12, 39, 14, 251, 8, 15, 251, 5, 2, 12, 38, 84, 80, 124, 12, 9, 23])
 list([1, 17, 6, 194, 337, 7, 4, 204, 22, 45, 254, 8, 106, 14, 123, 4, 2, 270, 2,
5, 2, 2, 732, 2098, 101, 405, 39, 14, 1034, 4, 1310, 9, 115, 50, 305, 12, 47, 4, 1
68, 5, 235, 7, 38, 111, 699, 102, 7, 4, 4039, 9245, 9, 24, 6, 78, 1099, 17, 2345,
2, 21, 27, 9685, 6139, 5, 2, 1603, 92, 1183, 4, 1310, 7, 4, 204, 42, 97, 90, 35, 2
21, 109, 29, 127, 27, 118, 8, 97, 12, 157, 21, 6789, 2, 9, 6, 66, 78, 1099, 4, 63
1, 1191, 5, 2642, 272, 191, 1070, 6, 7585, 8, 2197, 2, 2, 544, 5, 383, 1271, 848,
1468, 2, 497, 2, 8, 1597, 8778, 2, 21, 60, 27, 239, 9, 43, 8368, 209, 405, 10, 10,
12, 764, 40, 4, 248, 20, 12, 16, 5, 174, 1791, 72, 7, 51, 6, 1739, 22, 4, 204, 13
1, 9])] (25000,)
```

In [6]:

```
train_labels[0]
```

Out[6]:

```
1
```

In [7]:

```
len(train_labels)
```

Out[7]:

```
25000
```

In [8]:

```
len(train_labels)
```

Out[8]:

```
25000
```

In [9]:

```
test_labels[0]
```

Out[9]:

```
0
```

In [10]:

```
max([max(sequence) for sequence in test_data])
```

Out[10]:

```
9999
```

In [11]:

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

In [12]:

```
decoded_review
```

Out[12]:

```
"? this film was just brilliant casting location scenery story direction everyon
e's really suited the part they played and you could just imagine being there robe
rt ? is an amazing actor and now the same being director ? father came from the sa
me scottish island as myself so i loved the fact there was a real connection with
this film the witty remarks throughout the film were great it was just brilliant s
o much that i bought the film as soon as it was released for ? and would recommend
it to everyone to watch and the fly fishing was amazing really cried at the end it
was so sad and you know what they say if you cry at a film it must have been good
and this definitely was also ? to the two little boy's that played the ? of norman
and paul they were just brilliant children are often left out of the ? list i thin
k because the stars that play them all grown up are such a big profile for the who
le film but these children are amazing and should be praised for what they have do
ne don't you think the whole story was so lovely because it was true and was someo
ne's life after all that was shared with us all"
```

In [13]:

```
#Getting the Data Ready
```

In [14]:

```
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
```

In [15]:

```
#Vectorization of Data
x_train = vectorize_sequences(train_data)
x_test = vectorize_sequences(test_data)
```

In [16]:

```
x_train[0]
```

Out[16]:

```
array([0., 1., 1., ..., 0., 0., 0.])
```

In [17]:

```
x_test[0]
```

Out[17]:

```
array([0., 1., 1., ..., 0., 0., 0.])
```

In [18]:

```
#Vectorization of labels
y_train = np.asarray(train_labels).astype("float32")
y_test = np.asarray(test_labels).astype("float32")
```

In [19]:

```
#Constructing a model with relu and compiling it
#This is the most basic setup you will ever see: our labels are scalars (1 and 0s)
#At last, decide on an optimizer and a loss function. We should use the binary_cros
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")
])
```

```
WARNING:tensorflow:From C:\Users\Vijay\anaconda3\Lib\site-packages\keras\src\backe
nd.py:873: The name tf.get_default_graph is deprecated. Please use tf.compat.v1.ge
t_default_graph instead.
```

In [20]:

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

```
WARNING:tensorflow:From C:\Users\Vijay\anaconda3\Lib\site-packages\keras\src\optim
izers\__init__.py:309: The name tf.train.Optimizer is deprecated. Please use tf.co
mpat.v1.train.Optimizer instead.
```

In [21]:

```
#Verifying the methodology
```

In [22]:

```
#By deleting 10,000 samples from the initial training set, we will create a "valida
x_val = x_train[:10000]
partial_x_train = x_train[10000:]
y_val = y_train[:10000]
partial_y_train = y_train[10000:]
```

In [23]:

```
#We will now train our model in 512-sample mini-batches for 20 epochs (20 iteration
```

In [24]:

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

```
Epoch 1/20
WARNING:tensorflow:From C:\Users\Vijay\anaconda3\Lib\site-packages\keras\src\utils
\tf_utils.py:492: The name tf.ragged.RaggedTensorValue is deprecated. Please use t
f.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From C:\Users\Vijay\anaconda3\Lib\site-packages\keras\src\engin
e\base_layer_utils.py:384: The name tf.executing_eagerly_outside_functions is depr
ecated. Please use tf.compat.v1.executing_eagerly_outside_functions instead.

30/30 [==============================] - 2s 36ms/step - loss: 0.5429 - accuracy:
0.7807 - val_loss: 0.4225 - val_accuracy: 0.8618
Epoch 2/20
30/30 [==============================] - 0s 11ms/step - loss: 0.3475 - accuracy:
0.8930 - val_loss: 0.3283 - val_accuracy: 0.8799
Epoch 3/20
30/30 [==============================] - 0s 11ms/step - loss: 0.2615 - accuracy:
0.9137 - val_loss: 0.2914 - val_accuracy: 0.8877
Epoch 4/20
30/30 [==============================] - 0s 13ms/step - loss: 0.2121 - accuracy:
0.9306 - val_loss: 0.2771 - val_accuracy: 0.8881
Epoch 5/20
30/30 [==============================] - 0s 11ms/step - loss: 0.1777 - accuracy:
0.9428 - val_loss: 0.2752 - val_accuracy: 0.8877
Epoch 6/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1535 - accuracy:
0.9497 - val_loss: 0.2982 - val_accuracy: 0.8818
Epoch 7/20
30/30 [==============================] - 0s 11ms/step - loss: 0.1323 - accuracy:
0.9596 - val_loss: 0.2930 - val_accuracy: 0.8848
Epoch 8/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1146 - accuracy:
0.9665 - val_loss: 0.3011 - val_accuracy: 0.8833
Epoch 9/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0995 - accuracy: 0.
9706 - val_loss: 0.3391 - val_accuracy: 0.8779
Epoch 10/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0859 - accuracy: 0.
9766 - val_loss: 0.3223 - val_accuracy: 0.8821
Epoch 11/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0754 - accuracy: 0.
9806 - val_loss: 0.3623 - val_accuracy: 0.8745
Epoch 12/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0641 - accuracy: 0.
9843 - val_loss: 0.3613 - val_accuracy: 0.8785
Epoch 13/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0576 - accuracy: 0.
9861 - val_loss: 0.3737 - val_accuracy: 0.8793
Epoch 14/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0466 - accuracy: 0.
9905 - val_loss: 0.3926 - val_accuracy: 0.8766
Epoch 15/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0410 - accuracy: 0.
9917 - val_loss: 0.4145 - val_accuracy: 0.8751
Epoch 16/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0362 - accuracy: 0.
9926 - val_loss: 0.4470 - val_accuracy: 0.8706
Epoch 17/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0287 - accuracy: 0.
9954 - val_loss: 0.5052 - val_accuracy: 0.8683
Epoch 18/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0269 - accuracy: 0.
9956 - val_loss: 0.4841 - val_accuracy: 0.8726
Epoch 19/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0215 - accuracy: 0.
```

```
9967 - val_loss: 0.4960 - val_accuracy: 0.8729
Epoch 20/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0155 - accuracy: 0.
9990 - val_loss: 0.5526 - val_accuracy: 0.8617
```

In [25]:

```
#It is important to acknowledge that the model.fit() function generates a History c
```

In [26]:

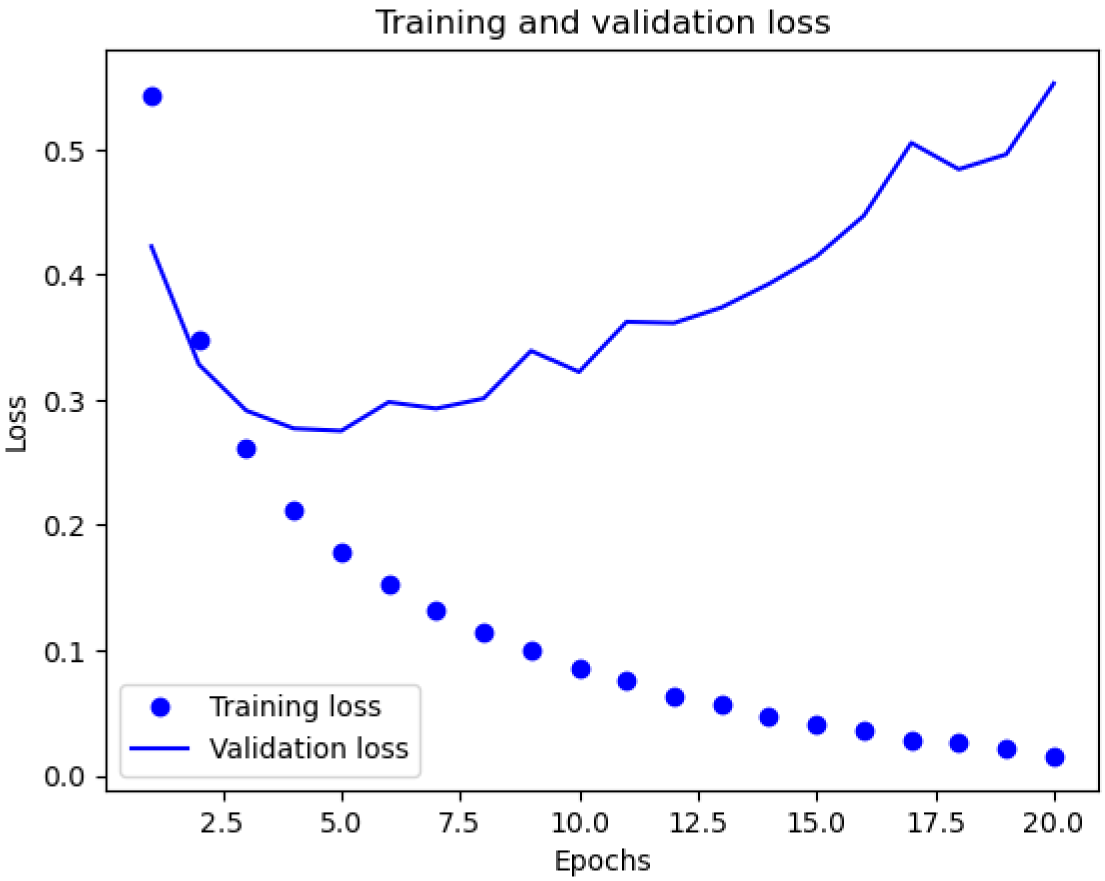
```
history_dict = history.history
history_dict.keys()
```

Out[26]:

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

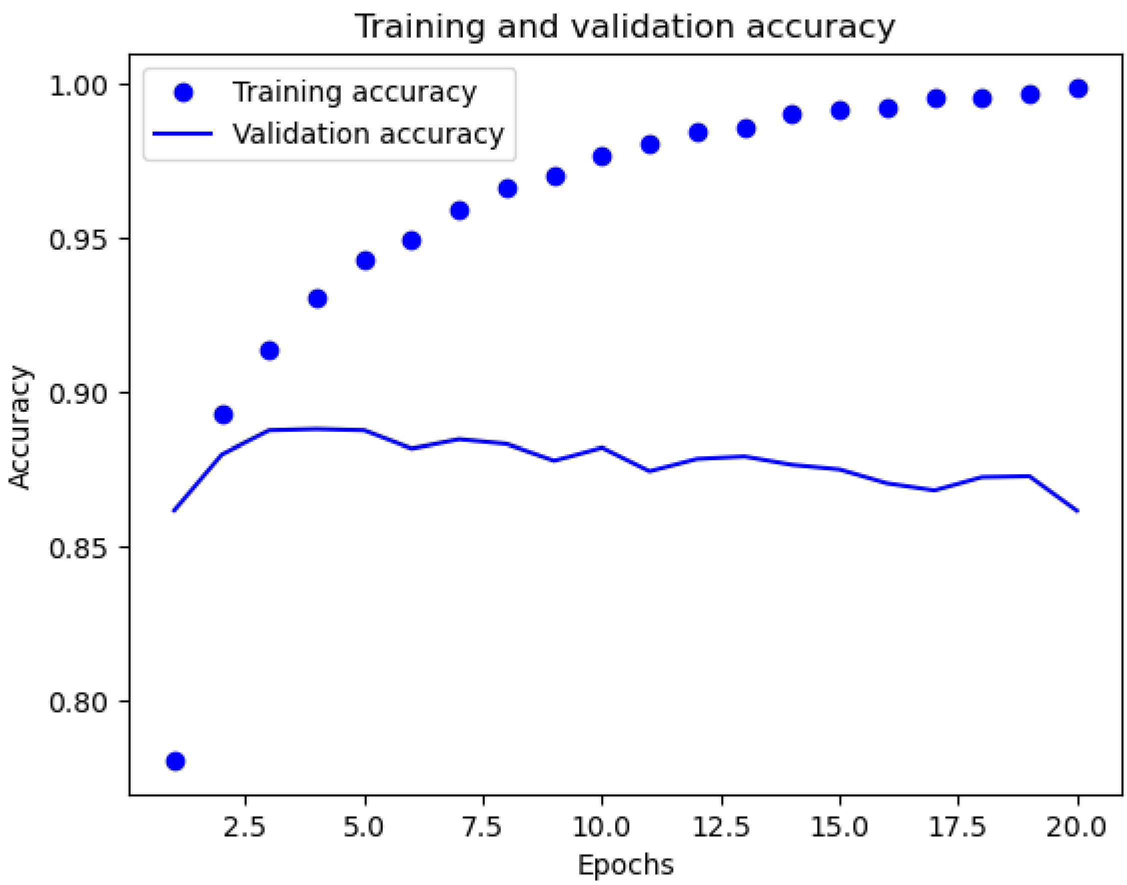
In [27]:

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



In [28]:

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



In [29]:
```
#Whereas the solid lines represent validation loss and accuracy, the dots represent
```

In [30]:
```
#Model retraining
```

In [31]:
```
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 7ms/step - loss: 0.4637 - accuracy: 0.
8155
Epoch 2/4
49/49 [==============================] - 0s 7ms/step - loss: 0.2703 - accuracy: 0.
9048
Epoch 3/4
49/49 [==============================] - 0s 7ms/step - loss: 0.2136 - accuracy: 0.
9237
Epoch 4/4
49/49 [==============================] - 0s 6ms/step - loss: 0.1805 - accuracy: 0.
9370
782/782 [==============================] - 1s 2ms/step - loss: 0.2849 - accuracy:
0.8864
```

In [32]:

```
results
```

Out[32]:

```
[0.28490492701530457, 0.8863599896430969]
```

In [33]:

```
#Accuracy of 88.5% on the test dataset. The loss value test is 0.2868
```

In [34]:

```
#Generating predictions about fresh data using a trained model
model.predict(x_test)
```

```
782/782 [==============================] - 1s 2ms/step
```

Out[34]:

```
array([[0.16948096],
       [0.9994732 ],
       [0.80792665],
       ...,
       [0.09143517],
       [0.07481366],
       [0.65304595]], dtype=float32)
```

In [35]:

```
#Two concealed layers were being used. To observe the impact on validation and test
#Experiment by using layers with varying numbers of concealed units: 32 pieces, 64
#In place of binary_crossentropy, try using the mse loss function.
#As an alternative to relu, consider using the tanh activation, which was well-like
#To improve your model's performance during validation, try implementing any of the
```

In [36]:

```
#Constructing a neural network with 1- hidden layer
```

In [37]:

```
model_1_layer = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])

model_1_layer.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])

x_val1 = x_train[:10000]
partial_x_train = x_train[10000:]

y_val1 = y_train[:10000]
partial_y_train = y_train[10000:]


history1_layer = model_1_layer.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val1, y_val1))
```

```
Epoch 1/20
30/30 [==============================] - 1s 24ms/step - loss: 0.5161 - accuracy:
0.7895 - val_loss: 0.4042 - val_accuracy: 0.8666
Epoch 2/20
30/30 [==============================] - 0s 9ms/step - loss: 0.3413 - accuracy: 0.
8927 - val_loss: 0.3474 - val_accuracy: 0.8657
Epoch 3/20
30/30 [==============================] - 0s 9ms/step - loss: 0.2696 - accuracy: 0.
9159 - val_loss: 0.2996 - val_accuracy: 0.8891
Epoch 4/20
30/30 [==============================] - 0s 9ms/step - loss: 0.2275 - accuracy: 0.
9260 - val_loss: 0.2838 - val_accuracy: 0.8906
Epoch 5/20
30/30 [==============================] - 0s 11ms/step - loss: 0.1978 - accuracy:
0.9383 - val_loss: 0.2840 - val_accuracy: 0.8860
Epoch 6/20
30/30 [==============================] - 0s 11ms/step - loss: 0.1751 - accuracy:
0.9463 - val_loss: 0.2750 - val_accuracy: 0.8883
Epoch 7/20
30/30 [==============================] - 0s 11ms/step - loss: 0.1592 - accuracy:
0.9503 - val_loss: 0.2814 - val_accuracy: 0.8878
Epoch 8/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1438 - accuracy:
0.9557 - val_loss: 0.2880 - val_accuracy: 0.8846
Epoch 9/20
30/30 [==============================] - 0s 12ms/step - loss: 0.1316 - accuracy:
0.9610 - val_loss: 0.2837 - val_accuracy: 0.8851
Epoch 10/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1192 - accuracy:
0.9663 - val_loss: 0.2905 - val_accuracy: 0.8856
Epoch 11/20
30/30 [==============================] - 0s 11ms/step - loss: 0.1106 - accuracy:
0.9687 - val_loss: 0.2963 - val_accuracy: 0.8841
Epoch 12/20
30/30 [==============================] - 0s 11ms/step - loss: 0.1001 - accuracy:
0.9743 - val_loss: 0.3108 - val_accuracy: 0.8830
Epoch 13/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0935 - accuracy:
0.9761 - val_loss: 0.3184 - val_accuracy: 0.8822
Epoch 14/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0859 - accuracy:
0.9794 - val_loss: 0.3212 - val_accuracy: 0.8818
Epoch 15/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0794 - accuracy:
0.9807 - val_loss: 0.3296 - val_accuracy: 0.8817
Epoch 16/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0737 - accuracy:
0.9836 - val_loss: 0.3355 - val_accuracy: 0.8816
Epoch 17/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0674 - accuracy:
0.9859 - val_loss: 0.3458 - val_accuracy: 0.8775
Epoch 18/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0628 - accuracy:
0.9873 - val_loss: 0.3527 - val_accuracy: 0.8792
Epoch 19/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0577 - accuracy:
0.9897 - val_loss: 0.3807 - val_accuracy: 0.8753
Epoch 20/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0541 - accuracy:
0.9902 - val_loss: 0.3717 - val_accuracy: 0.8784
```

In [38]:

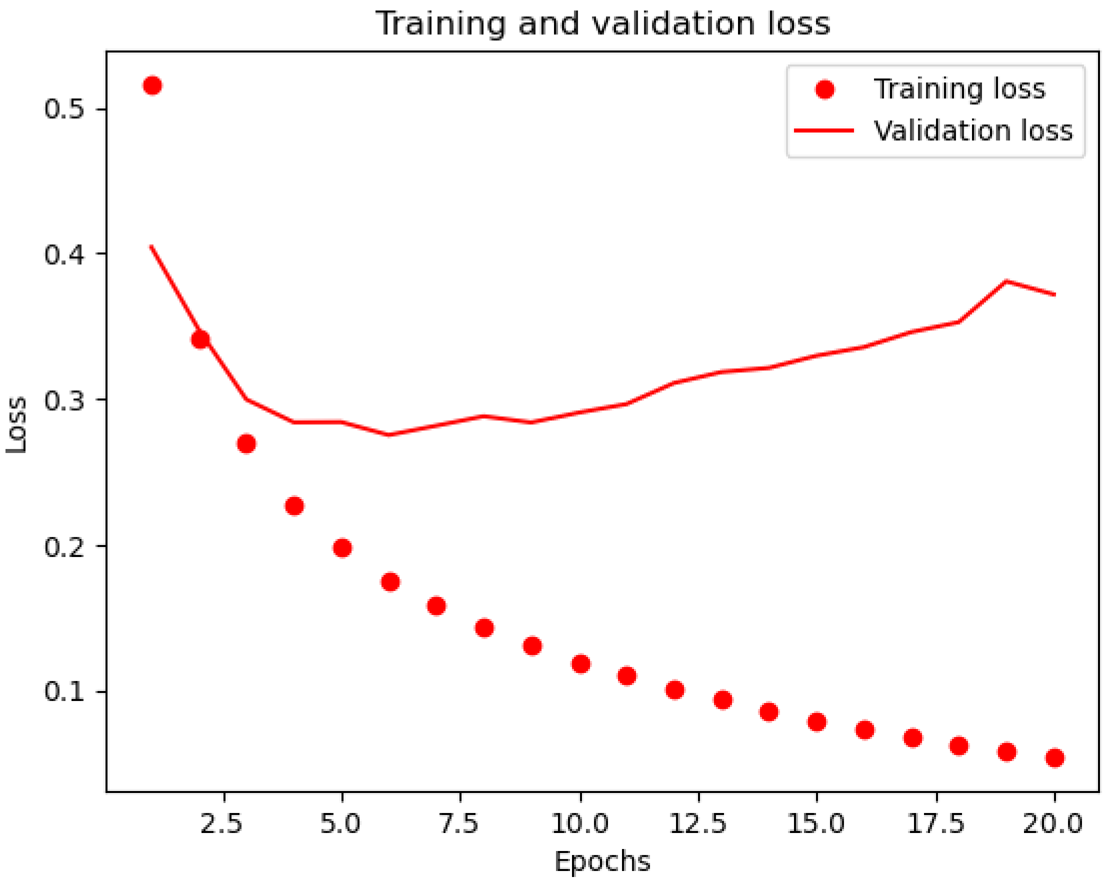
```
history_dict1 = history1_layer.history
history_dict1.keys()
```

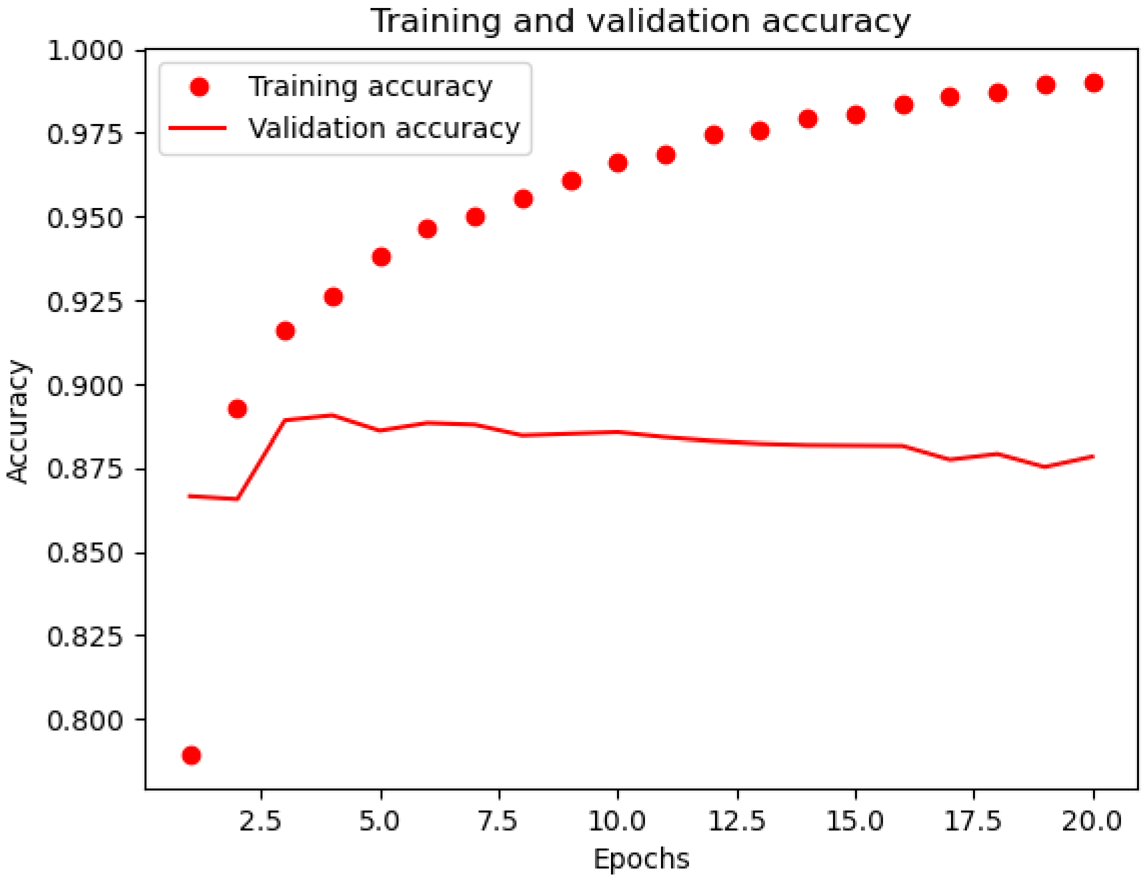
Out[38]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [39]:
```
import matplotlib.pyplot as plt
history_dict1 = history1_layer.history
loss_value1 = history_dict1["loss"]
val_loss_value1 = history_dict1["val_loss"]
epochs1 = range(1, len(loss_value1) + 1)

#Plotting graph of Training against Validation loss
plt.plot(epochs1, loss_value1, "ro", label="Training loss")
plt.plot(epochs1, val_loss_value1, "r", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()

#Plotting graph of Training against Validation Accuracy
plt.clf()
accuracy1 = history_dict1["accuracy"]
val_accuracy1 = history_dict1["val_accuracy"]
plt.plot(epochs1, accuracy1, "ro", label="Training accuracy")
plt.plot(epochs1, val_accuracy1, "r", label="Validation accuracy")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```





In [40]:
```
#It is evident that the model with fewer layers starts to overfit later than the re
```

In [41]:
```
#Generating model
model_1_layer = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])

model_1_layer.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model_1_layer.fit(x_train, y_train, epochs=5, batch_size=512)
result_1_layer = model_1_layer.evaluate(x_test, y_test)
```

```
Epoch 1/5
49/49 [==============================] - 1s 8ms/step - loss: 0.4510 - accuracy: 0.
8198
Epoch 2/5
49/49 [==============================] - 0s 8ms/step - loss: 0.2833 - accuracy: 0.
9020
Epoch 3/5
49/49 [==============================] - 0s 7ms/step - loss: 0.2275 - accuracy: 0.
9208
Epoch 4/5
49/49 [==============================] - 0s 7ms/step - loss: 0.1992 - accuracy: 0.
9306
Epoch 5/5
49/49 [==============================] - 0s 6ms/step - loss: 0.1776 - accuracy: 0.
9391
782/782 [==============================] - 1s 2ms/step - loss: 0.2839 - accuracy:
0.8848
```

In [42]:
```
print(result_1_layer)
```

```
[0.28392335772514343, 0.8848000168800354]
```

In [43]:

```
#The loss on the test set is 0.279, and the accuracy is 88.67%.
```

In [44]:

```
model_1_layer.predict(x_test)
```

```
782/782 [==============================] - 1s 2ms/step
```

Out[44]:

```
array([[0.18286084],
       [0.9991824 ],
       [0.7331096 ],
       ...,
       [0.10014362],
       [0.0700555 ],
       [0.4963487 ]], dtype=float32)
```

In [45]:

```
#Building a neural network with 3-hidden layers
```

In [46]:

```
model_3_layers = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
model_3_layers.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
x_val3 = x_train[:10000]
partial_x_train = x_train[10000:]

y_val3 = y_train[:10000]
partial_y_train = y_train[10000:]

history_3_layers = model_3_layers.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val3, y_val3))
```

```
Epoch 1/20
30/30 [==============================] - 2s 39ms/step - loss: 0.5398 - accuracy:
0.7679 - val_loss: 0.3874 - val_accuracy: 0.8687
Epoch 2/20
30/30 [==============================] - 0s 10ms/step - loss: 0.3187 - accuracy:
0.8903 - val_loss: 0.3094 - val_accuracy: 0.8843
Epoch 3/20
30/30 [==============================] - 0s 9ms/step - loss: 0.2343 - accuracy: 0.
9212 - val_loss: 0.2800 - val_accuracy: 0.8919
Epoch 4/20
30/30 [==============================] - 0s 9ms/step - loss: 0.1847 - accuracy: 0.
9383 - val_loss: 0.2978 - val_accuracy: 0.8801
Epoch 5/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1501 - accuracy:
0.9509 - val_loss: 0.2898 - val_accuracy: 0.8861
Epoch 6/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1270 - accuracy:
0.9593 - val_loss: 0.3294 - val_accuracy: 0.8738
Epoch 7/20
30/30 [==============================] - 0s 9ms/step - loss: 0.1021 - accuracy: 0.
9685 - val_loss: 0.4224 - val_accuracy: 0.8543
Epoch 8/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0880 - accuracy:
0.9738 - val_loss: 0.3650 - val_accuracy: 0.8723
Epoch 9/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0695 - accuracy:
0.9809 - val_loss: 0.3584 - val_accuracy: 0.8811
Epoch 10/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0702 - accuracy:
0.9787 - val_loss: 0.3816 - val_accuracy: 0.8776
Epoch 11/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0409 - accuracy: 0.
9911 - val_loss: 0.4234 - val_accuracy: 0.8747
Epoch 12/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0427 - accuracy: 0.
9888 - val_loss: 0.4326 - val_accuracy: 0.8734
Epoch 13/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0376 - accuracy: 0.
9904 - val_loss: 0.4619 - val_accuracy: 0.8722
Epoch 14/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0287 - accuracy:
0.9924 - val_loss: 0.4800 - val_accuracy: 0.8736
Epoch 15/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0315 - accuracy:
0.9918 - val_loss: 0.5065 - val_accuracy: 0.8715
Epoch 16/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0109 - accuracy:
0.9993 - val_loss: 0.5280 - val_accuracy: 0.8716
Epoch 17/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0218 - accuracy: 0.
9942 - val_loss: 0.5528 - val_accuracy: 0.8743
Epoch 18/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0072 - accuracy: 0.
9996 - val_loss: 0.7821 - val_accuracy: 0.8418
Epoch 19/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0136 - accuracy: 0.
9970 - val_loss: 0.6018 - val_accuracy: 0.8712
Epoch 20/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0199 - accuracy:
0.9933 - val_loss: 0.6279 - val_accuracy: 0.8713
```

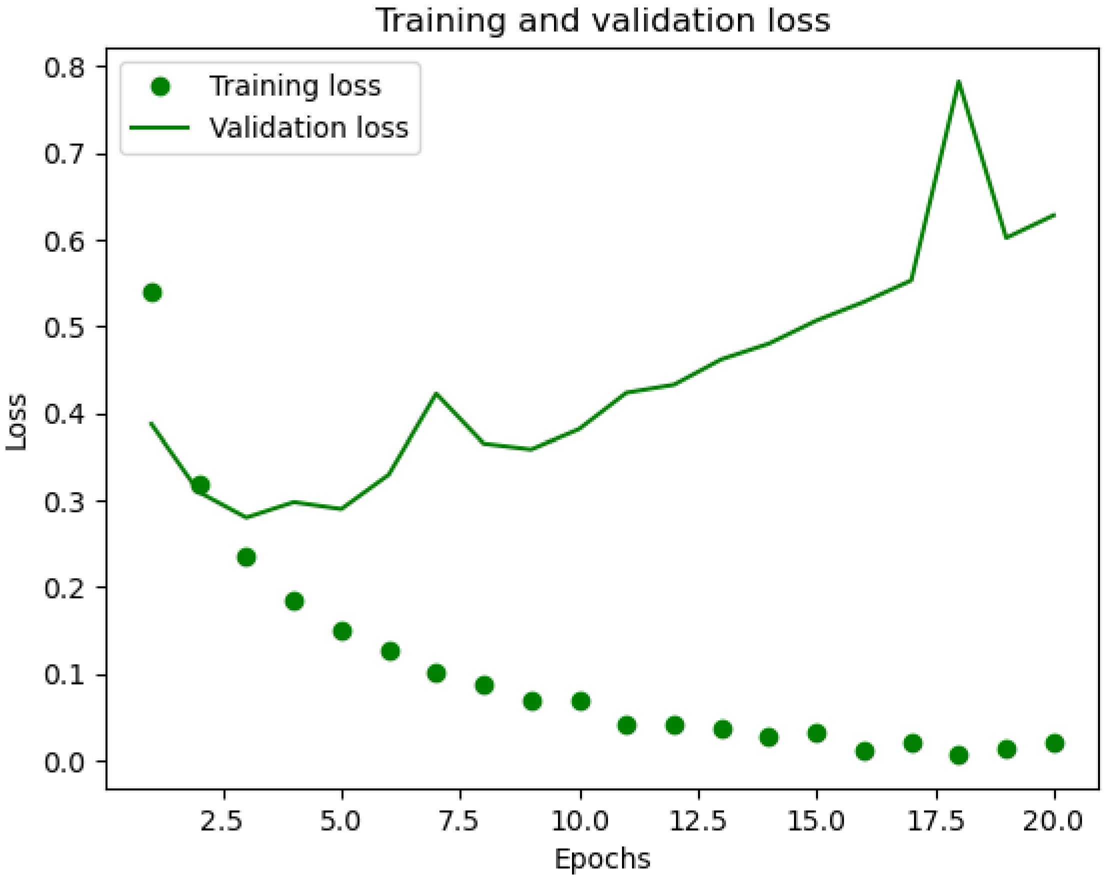
In [47]:

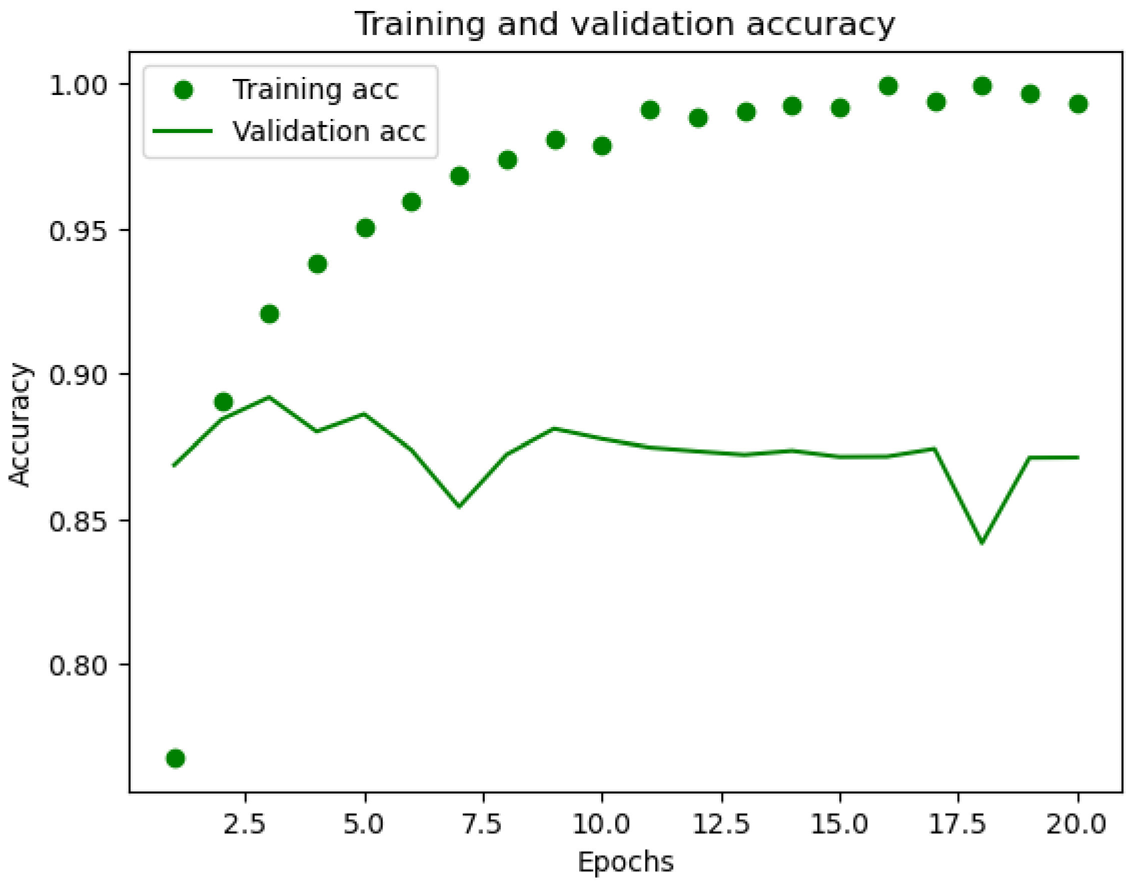
```
history_dict_3 = history_3_layers.history
history_dict_3.keys()
```

```
Out[47]: dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

```
In [48]: loss_val3 = history_dict_3["loss"]
         val_loss_val3 = history_dict_3["val_loss"]
         epochs3 = range(1, len(loss_val3) + 1)
         plt.plot(epochs3, loss_val3, "go", label="Training loss")
         plt.plot(epochs3, val_loss_val3, "g", label="Validation loss")
         plt.title("Training and validation loss")
         plt.xlabel("Epochs")
         plt.ylabel("Loss")
         plt.legend()
         plt.show()


         plt.clf()
         accuracy3 = history_dict_3["accuracy"]
         val_accuracy3 = history_dict_3["val_accuracy"]
         plt.plot(epochs3, accuracy3, "go", label="Training acc")
         plt.plot(epochs3, val_accuracy3, "g", label="Validation acc")
         plt.title("Training and validation accuracy")
         plt.xlabel("Epochs")
         plt.ylabel("Accuracy")
         plt.legend()
         plt.show()
```





In [49]:

```
#As we can see, overfitting occurs with more layers, so let's use three epochs.
```

In [50]:

```
model_3_layers = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])


model_3_layers.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])

model_3_layers.fit(x_train, y_train, epochs=3, batch_size=512)
results_3_layers = model_3_layers.evaluate(x_test, y_test)
```

```
Epoch 1/3
49/49 [==============================] - 1s 7ms/step - loss: 0.4880 - accuracy: 0.
7989
Epoch 2/3
49/49 [==============================] - 0s 6ms/step - loss: 0.2721 - accuracy: 0.
9019
Epoch 3/3
49/49 [==============================] - 0s 6ms/step - loss: 0.2109 - accuracy: 0.
9238
782/782 [==============================] - 2s 2ms/step - loss: 0.3292 - accuracy:
0.8677
```

In [51]:

```
print(results_3_layers)
```

```
[0.32924187183380127, 0.8677200078964233]
```

In [52]:

```
model_3_layers.predict(x_test)
```

Out[52]:

```
782/782 [==============================] - 2s 2ms/step
array([[0.17476751],
       [0.9966734 ],
       [0.4132907 ],
       ...,
       [0.08154602],
       [0.04734027],
       [0.31040075]], dtype=float32)
```

In [53]:

```
#The number of layers in the model does not greatly increase its accuracy. But the
```

In [54]:

```
#Building Neural Network with 32 Hidden units & 3 layers.
```

In [55]:

```
model_32_units = keras.Sequential([
    layers.Dense(32, activation="relu"),
    layers.Dense(32, activation="relu"),
    layers.Dense(32, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
#model compilation
model_32_units.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
#model validation
x_val_32 = x_train[:10000]
partial_x_train = x_train[10000:]

y_val_32 = y_train[:10000]
partial_y_train = y_train[10000:]


history_32_units = model_32_units.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val_32, y_val_32))
```

```
Epoch 1/20
30/30 [==============================] - 2s 42ms/step - loss: 0.5195 - accuracy:
0.7551 - val_loss: 0.3628 - val_accuracy: 0.8667
Epoch 2/20
30/30 [==============================] - 0s 12ms/step - loss: 0.3012 - accuracy:
0.8857 - val_loss: 0.2935 - val_accuracy: 0.8849
Epoch 3/20
30/30 [==============================] - 0s 11ms/step - loss: 0.2215 - accuracy:
0.9209 - val_loss: 0.2805 - val_accuracy: 0.8885
Epoch 4/20
30/30 [==============================] - 0s 13ms/step - loss: 0.1845 - accuracy:
0.9317 - val_loss: 0.3076 - val_accuracy: 0.8779
Epoch 5/20
30/30 [==============================] - 0s 11ms/step - loss: 0.1450 - accuracy:
0.9479 - val_loss: 0.3038 - val_accuracy: 0.8810
Epoch 6/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1227 - accuracy:
0.9553 - val_loss: 0.3154 - val_accuracy: 0.8856
Epoch 7/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0990 - accuracy:
0.9660 - val_loss: 0.3391 - val_accuracy: 0.8786
Epoch 8/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0764 - accuracy:
0.9763 - val_loss: 0.3965 - val_accuracy: 0.8672
Epoch 9/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0634 - accuracy:
0.9808 - val_loss: 0.5938 - val_accuracy: 0.8460
Epoch 10/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0487 - accuracy:
0.9854 - val_loss: 0.4986 - val_accuracy: 0.8556
Epoch 11/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0370 - accuracy:
0.9907 - val_loss: 0.5867 - val_accuracy: 0.8596
Epoch 12/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0217 - accuracy:
0.9961 - val_loss: 0.4833 - val_accuracy: 0.8771
Epoch 13/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0357 - accuracy:
0.9894 - val_loss: 0.5087 - val_accuracy: 0.8760
Epoch 14/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0393 - accuracy:
0.9889 - val_loss: 0.5346 - val_accuracy: 0.8752
Epoch 15/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0069 - accuracy:
0.9995 - val_loss: 0.5587 - val_accuracy: 0.8739
Epoch 16/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0436 - accuracy:
0.9885 - val_loss: 0.5764 - val_accuracy: 0.8735
Epoch 17/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0043 - accuracy:
0.9997 - val_loss: 0.6065 - val_accuracy: 0.8762
Epoch 18/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0029 - accuracy:
0.9999 - val_loss: 0.6677 - val_accuracy: 0.8753
Epoch 19/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0341 - accuracy:
0.9910 - val_loss: 0.6990 - val_accuracy: 0.8745
Epoch 20/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0014 - accuracy:
0.9999 - val_loss: 0.7165 - val_accuracy: 0.8735
```

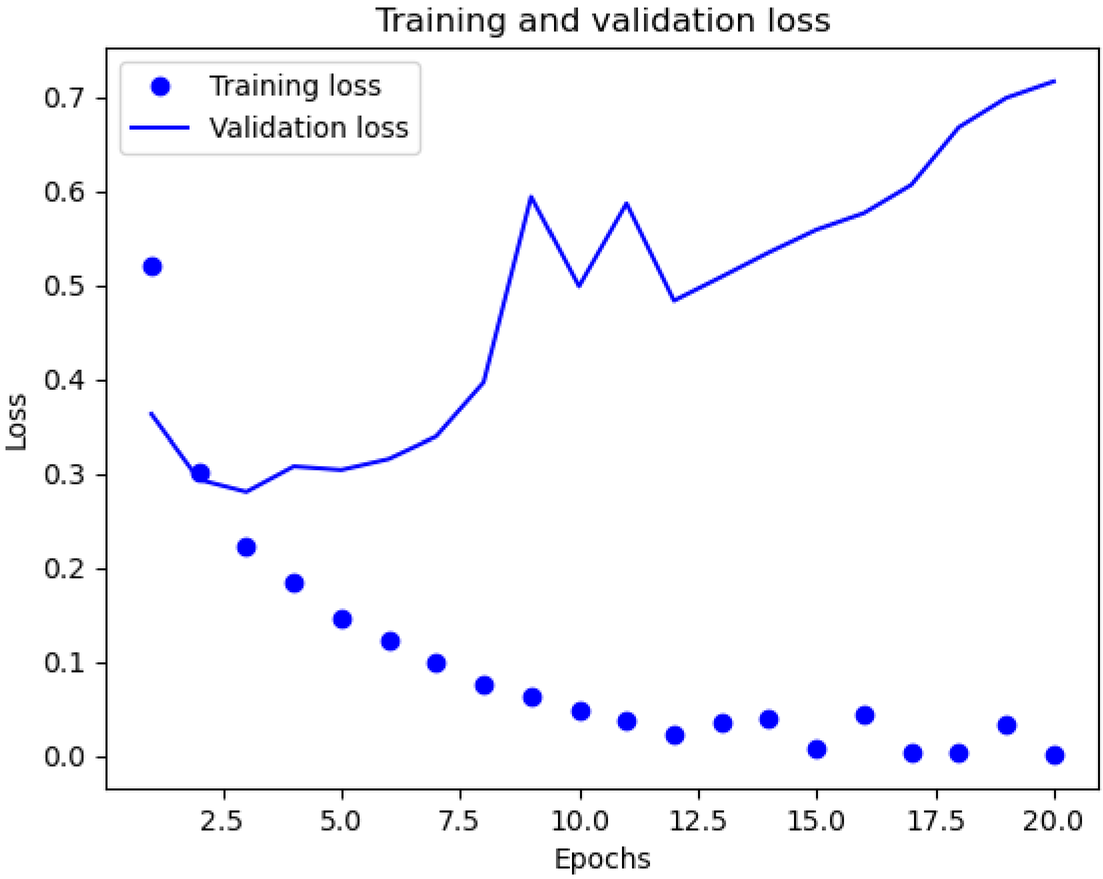
In [56]:

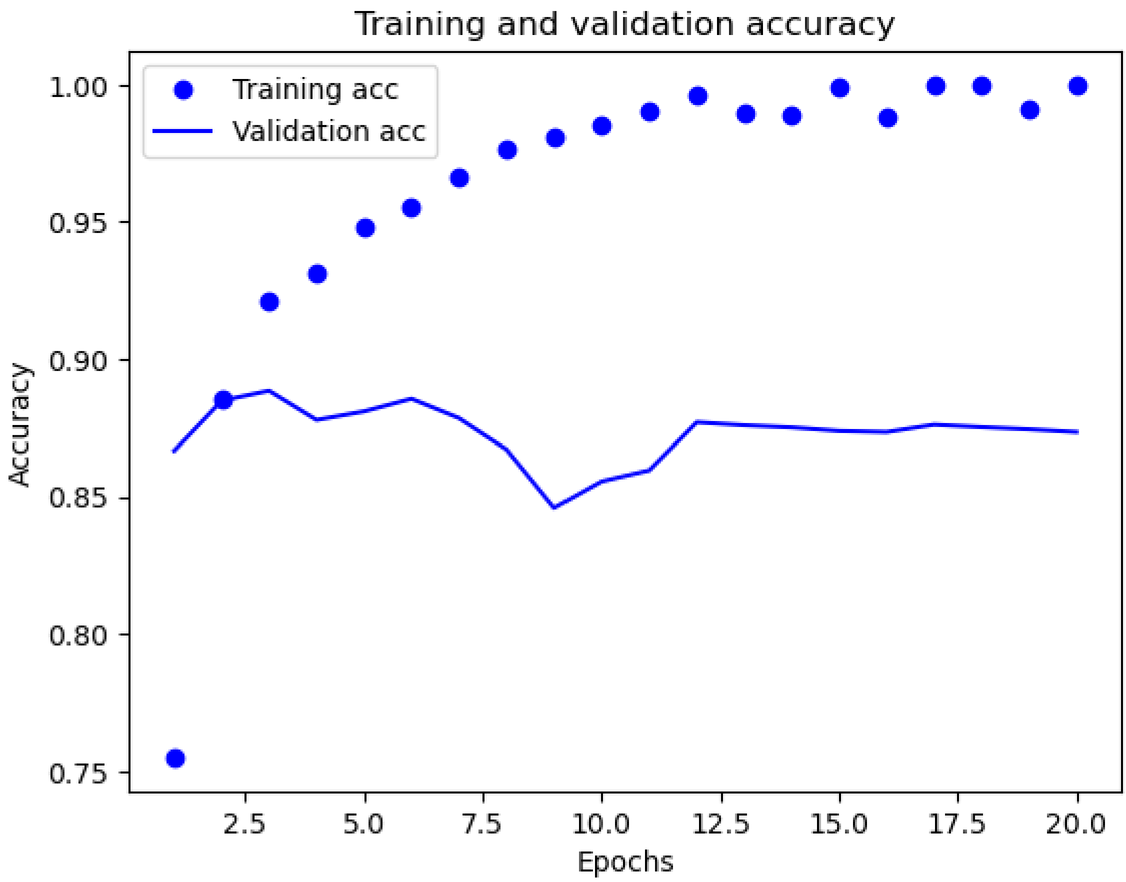
```
history_dict_32 = history_32_units.history
history_dict_32.keys()
```

Out[56]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [57]:
```
loss_value_32 = history_dict_32["loss"]
val_loss_value_32 = history_dict_32["val_loss"]
epochs_32 = range(1, len(loss_value_32) + 1)
plt.plot(epochs_32, loss_value_32, "bo", label="Training loss")
plt.plot(epochs_32, val_loss_value_32, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()


plt.clf()
accuracy_32 = history_dict_32["accuracy"]
val_accuracy_32 = history_dict_32["val_accuracy"]
plt.plot(epochs_32, accuracy_32, "bo", label="Training acc")
plt.plot(epochs_32, val_accuracy_32, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```





In [58]:
```
history_32_units = model_32_units.fit(x_train, y_train, epochs=3, batch_size=512)
results_32_units = model_32_units.evaluate(x_test, y_test)
results_32_units
```

```
Epoch 1/3
49/49 [==============================] - 0s 9ms/step - loss: 0.2004 - accuracy: 0.
9459
Epoch 2/3
49/49 [==============================] - 0s 9ms/step - loss: 0.1105 - accuracy: 0.
9646
Epoch 3/3
49/49 [==============================] - 0s 8ms/step - loss: 0.0715 - accuracy: 0.
9787
782/782 [==============================] - 2s 2ms/step - loss: 0.4224 - accuracy:
0.8679
```

Out[58]:
```
[0.42241066694259644, 0.8678799867630005]
```

In [59]:
```
model_32_units.predict(x_test)
```

```
782/782 [==============================] - 1s 2ms/step
```

Out[59]:
```
array([[0.09049544],
       [0.9999985 ],
       [0.9742518 ],
       ...,
       [0.17571723],
       [0.02408249],
       [0.9263707 ]], dtype=float32)
```

In [60]:
```
#validation set accuracy = 85.7%
```

In [61]:
```
#Having the model with 64 units and 2 layers.
```

In [62]:
```
model_64_units = keras.Sequential([
    layers.Dense(64, activation="relu"),
    layers.Dense(64, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
model_64_units.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
#Validation

x_val_64 = x_train[:10000]
partial_x_train = x_train[10000:]

y_val_64 = y_train[:10000]
partial_y_train = y_train[10000:]

history_64 = model_64_units.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val_64, y_val_64))
```

```
Epoch 1/20
30/30 [==============================] - 2s 44ms/step - loss: 0.5222 - accuracy:
0.7553 - val_loss: 0.3936 - val_accuracy: 0.8355
Epoch 2/20
30/30 [==============================] - 1s 17ms/step - loss: 0.3066 - accuracy:
0.8840 - val_loss: 0.3060 - val_accuracy: 0.8753
Epoch 3/20
30/30 [==============================] - 0s 16ms/step - loss: 0.2338 - accuracy:
0.9097 - val_loss: 0.2838 - val_accuracy: 0.8853
Epoch 4/20
30/30 [==============================] - 0s 16ms/step - loss: 0.1889 - accuracy:
0.9311 - val_loss: 0.2775 - val_accuracy: 0.8895
Epoch 5/20
30/30 [==============================] - 0s 16ms/step - loss: 0.1579 - accuracy:
0.9429 - val_loss: 0.3303 - val_accuracy: 0.8743
Epoch 6/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1269 - accuracy:
0.9570 - val_loss: 0.3268 - val_accuracy: 0.8754
Epoch 7/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1141 - accuracy:
0.9605 - val_loss: 0.3212 - val_accuracy: 0.8852
Epoch 8/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0867 - accuracy:
0.9713 - val_loss: 0.3962 - val_accuracy: 0.8636
Epoch 9/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0737 - accuracy:
0.9759 - val_loss: 0.3638 - val_accuracy: 0.8820
Epoch 10/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0588 - accuracy:
0.9820 - val_loss: 0.7155 - val_accuracy: 0.8030
Epoch 11/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0474 - accuracy:
0.9870 - val_loss: 0.4442 - val_accuracy: 0.8750
Epoch 12/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0429 - accuracy:
0.9873 - val_loss: 0.4886 - val_accuracy: 0.8724
Epoch 13/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0362 - accuracy:
0.9887 - val_loss: 0.4592 - val_accuracy: 0.8773
Epoch 14/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0381 - accuracy:
0.9889 - val_loss: 0.4785 - val_accuracy: 0.8760
Epoch 15/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0095 - accuracy:
0.9995 - val_loss: 0.9543 - val_accuracy: 0.8209
Epoch 16/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0296 - accuracy:
0.9907 - val_loss: 0.5406 - val_accuracy: 0.8751
Epoch 17/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0334 - accuracy:
0.9892 - val_loss: 0.5309 - val_accuracy: 0.8760
Epoch 18/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0044 - accuracy:
0.9999 - val_loss: 0.5687 - val_accuracy: 0.8775
Epoch 19/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0335 - accuracy:
0.9903 - val_loss: 0.5830 - val_accuracy: 0.8767
Epoch 20/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0031 - accuracy:
0.9999 - val_loss: 0.6026 - val_accuracy: 0.8767
```

In [63]:

```
history_dict_64 = history_64.history
history_dict_64.keys()
```

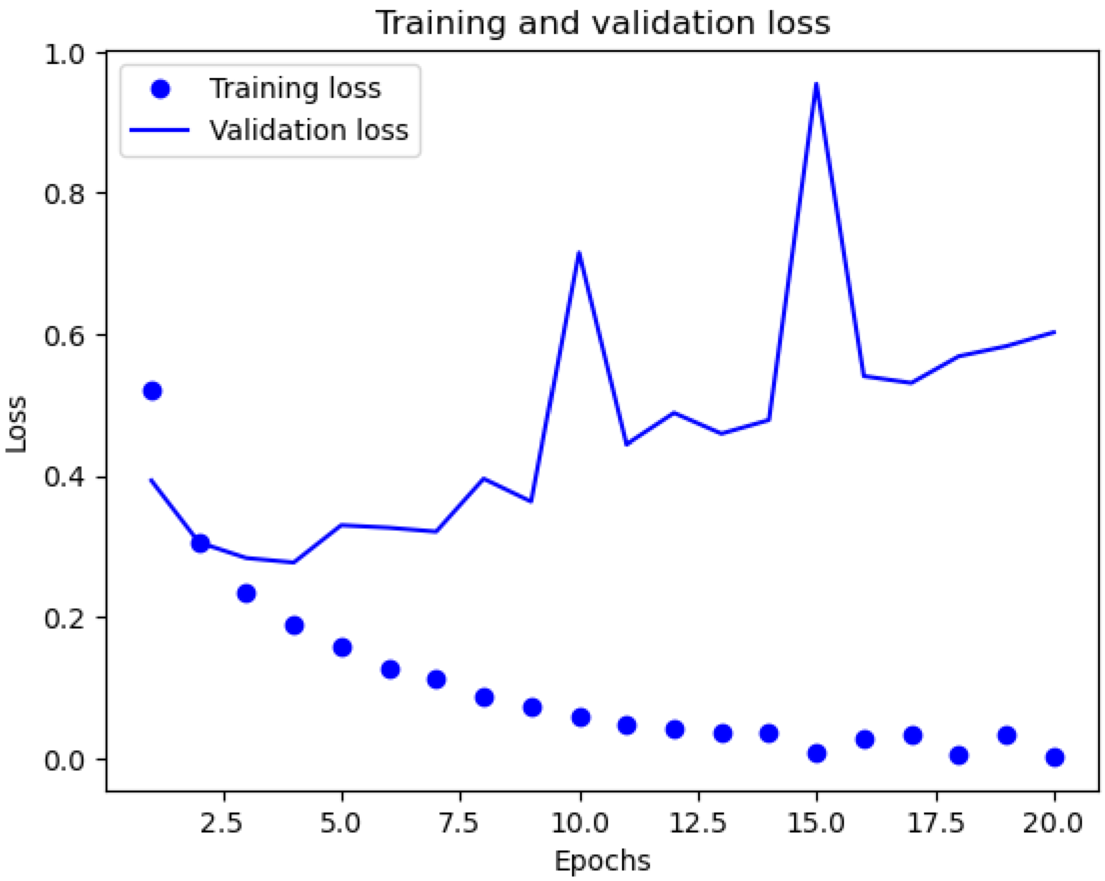
Out[63]:

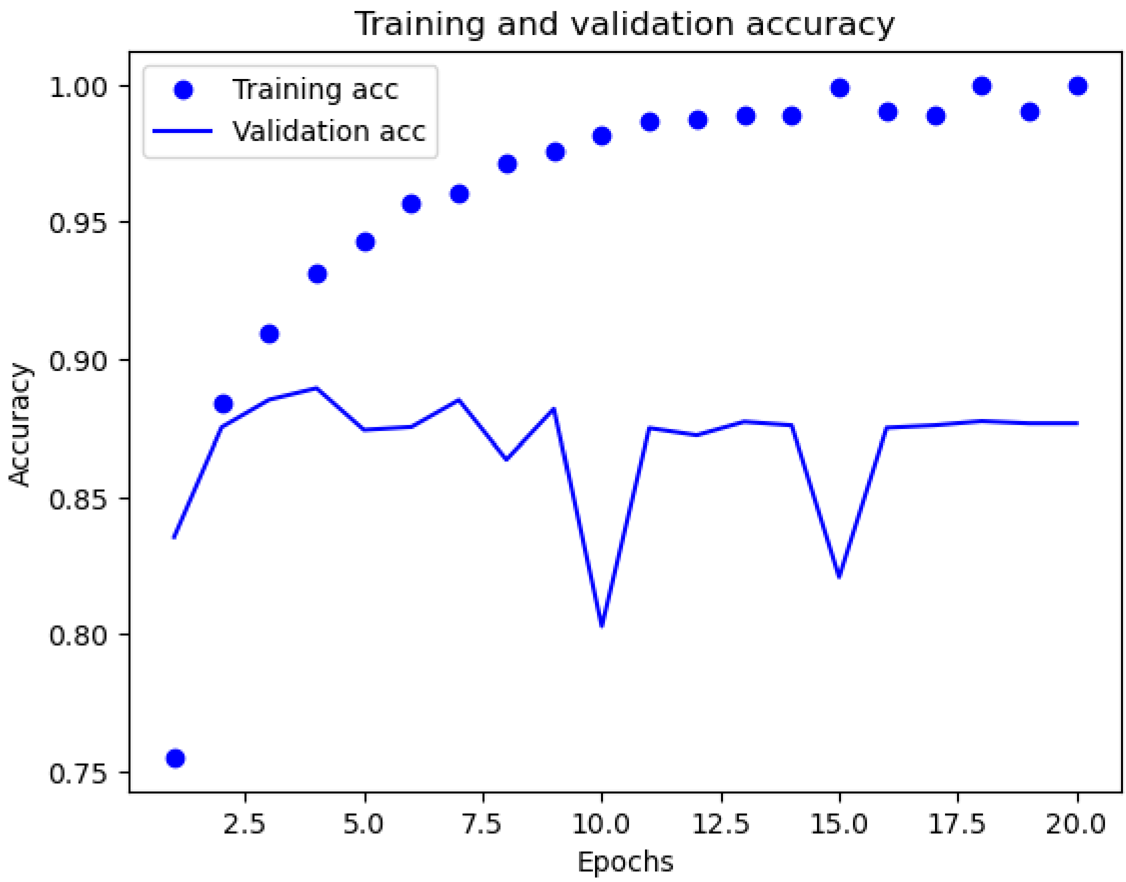
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [64]:

```
loss_value64 = history_dict_64["loss"]
val_loss_value64 = history_dict_64["val_loss"]
epochs_64 = range(1, len(loss_value64) + 1)
plt.plot(epochs_64, loss_value64, "bo", label="Training loss")
plt.plot(epochs_64, val_loss_value64, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()

plt.clf()
accuracy_64 = history_dict_64["accuracy"]
val_accuracy_64 = history_dict_64["val_accuracy"]
plt.plot(epochs_64, accuracy_64, "bo", label="Training acc")
plt.plot(epochs_64, val_accuracy_64, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```





In [65]:

```
history_64 = model_64_units.fit(x_train, y_train, epochs=3, batch_size=512)
results_64_units = model_64_units.evaluate(x_test, y_test)
results_64_units
```

```
Epoch 1/3
49/49 [==============================] - 1s 12ms/step - loss: 0.1761 - accuracy:
0.9483
Epoch 2/3
49/49 [==============================] - 1s 12ms/step - loss: 0.1012 - accuracy:
0.9680
Epoch 3/3
49/49 [==============================] - 1s 12ms/step - loss: 0.0635 - accuracy:
0.9815
782/782 [==============================] - 2s 2ms/step - loss: 0.4181 - accuracy:
0.8699
```

Out[65]:

```
[0.4181140959262848, 0.869920015335083]
```

In [66]:

```
model_64_units.predict(x_test)
```

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

Out[66]:

```
array([[0.00998912],
       [0.9999998 ],
       [0.33275568],
       ...,
       [0.0133983 ],
       [0.00538828],
       [0.87959373]], dtype=float32)
```

In [67]:

```
#validation set accuracy = 86.92%
```

In [68]:

```
#Training the model with 128 units and 3 layers
```

In [69]:

```
model_128units = keras.Sequential([
    layers.Dense(128, activation="relu"),
    layers.Dense(128, activation="relu"),
    layers.Dense(128, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
model_128units.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
#Validation

x_val_128 = x_train[:10000]
partial_x_train = x_train[10000:]

y_val_128 = y_train[:10000]
partial_y_train = y_train[10000:]

history_128 = model_128units.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val_128, y_val_128))
```

```
Epoch 1/20
30/30 [==============================] - 2s 42ms/step - loss: 0.5388 - accuracy:
0.7334 - val_loss: 0.3390 - val_accuracy: 0.8680
Epoch 2/20
30/30 [==============================] - 1s 25ms/step - loss: 0.3176 - accuracy:
0.8760 - val_loss: 0.2861 - val_accuracy: 0.8849
Epoch 3/20
30/30 [==============================] - 1s 25ms/step - loss: 0.2256 - accuracy:
0.9123 - val_loss: 0.2851 - val_accuracy: 0.8832
Epoch 4/20
30/30 [==============================] - 1s 25ms/step - loss: 0.1833 - accuracy:
0.9300 - val_loss: 0.3073 - val_accuracy: 0.8715
Epoch 5/20
30/30 [==============================] - 1s 25ms/step - loss: 0.1308 - accuracy:
0.9521 - val_loss: 0.3157 - val_accuracy: 0.8785
Epoch 6/20
30/30 [==============================] - 1s 26ms/step - loss: 0.1220 - accuracy:
0.9547 - val_loss: 0.3043 - val_accuracy: 0.8839
Epoch 7/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0779 - accuracy:
0.9749 - val_loss: 0.3600 - val_accuracy: 0.8827
Epoch 8/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0757 - accuracy:
0.9782 - val_loss: 0.3819 - val_accuracy: 0.8810
Epoch 9/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0344 - accuracy:
0.9906 - val_loss: 0.6153 - val_accuracy: 0.8429
Epoch 10/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0139 - accuracy:
0.9971 - val_loss: 0.4912 - val_accuracy: 0.8791
Epoch 11/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0187 - accuracy:
0.9963 - val_loss: 0.9440 - val_accuracy: 0.8020
Epoch 12/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0139 - accuracy:
0.9953 - val_loss: 0.5496 - val_accuracy: 0.8754
Epoch 13/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0014 - accuracy:
0.9999 - val_loss: 0.6445 - val_accuracy: 0.8768
Epoch 14/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0996 - accuracy:
0.9819 - val_loss: 0.5620 - val_accuracy: 0.8751
Epoch 15/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0017 - accuracy:
1.0000 - val_loss: 0.5947 - val_accuracy: 0.8765
Epoch 16/20
30/30 [==============================] - 1s 25ms/step - loss: 7.6445e-04 - accurac
y: 1.0000 - val_loss: 0.6635 - val_accuracy: 0.8765
Epoch 17/20
30/30 [==============================] - 1s 25ms/step - loss: 3.7534e-04 - accurac
y: 1.0000 - val_loss: 0.7390 - val_accuracy: 0.8772
Epoch 18/20
30/30 [==============================] - 1s 25ms/step - loss: 2.0022e-04 - accurac
y: 1.0000 - val_loss: 0.7970 - val_accuracy: 0.8763
Epoch 19/20
30/30 [==============================] - 1s 27ms/step - loss: 1.2699e-04 - accurac
y: 1.0000 - val_loss: 0.8390 - val_accuracy: 0.8760
Epoch 20/20
30/30 [==============================] - 1s 24ms/step - loss: 8.7843e-05 - accurac
y: 1.0000 - val_loss: 0.8685 - val_accuracy: 0.8761
```

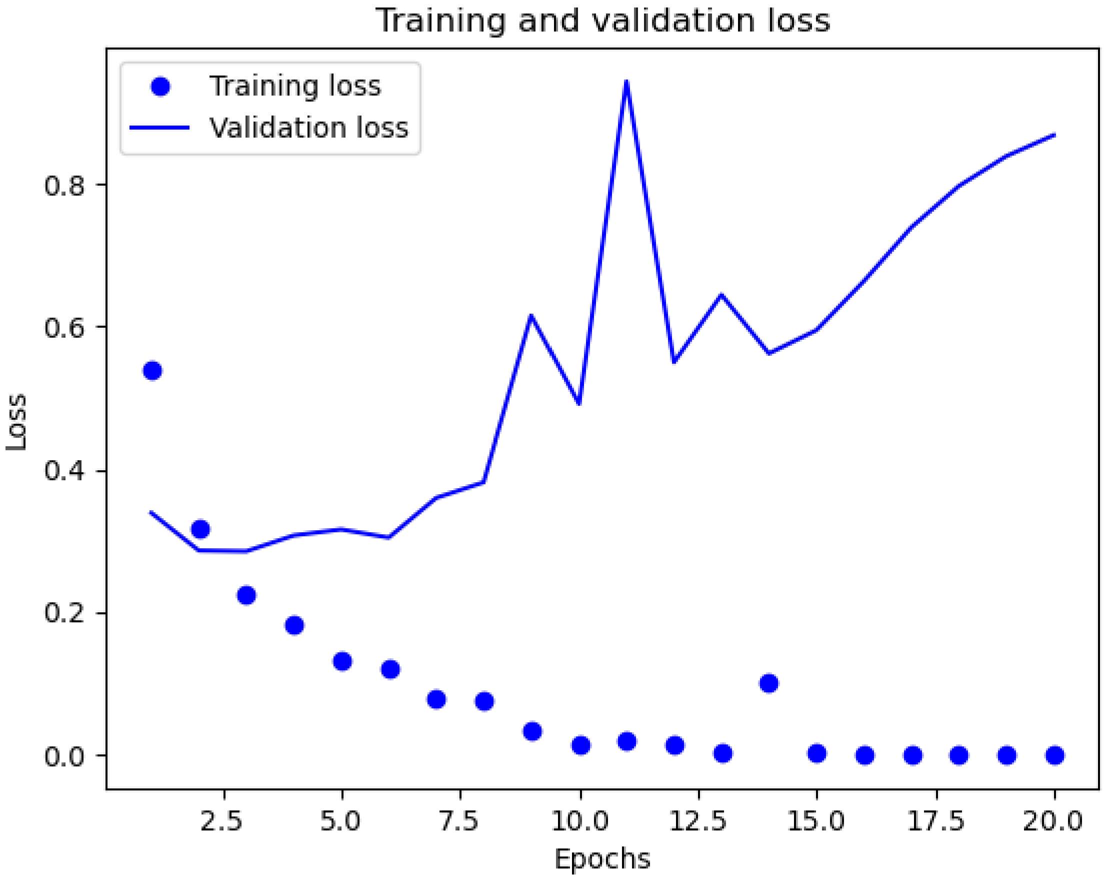
In [70]:

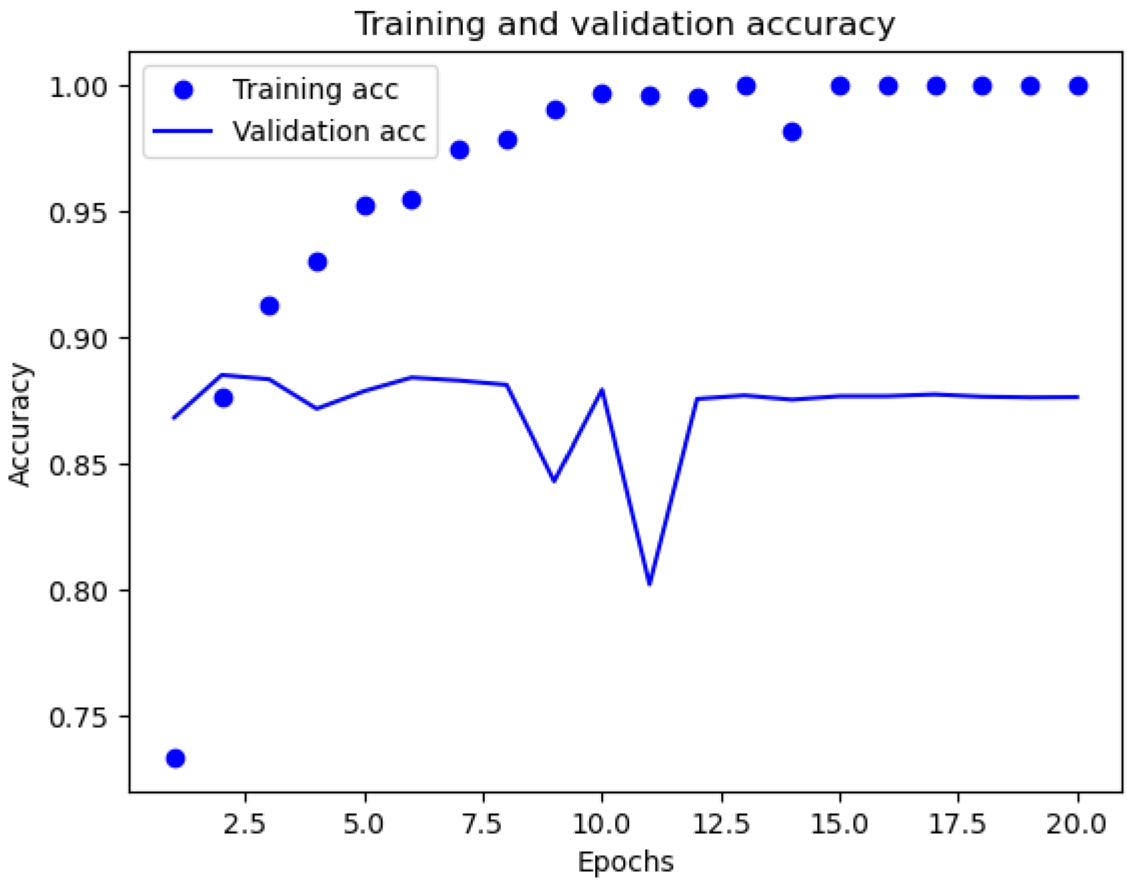
```
history_dict_128 = history_128.history
history_dict_128.keys()
```

```
Out[70]: dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

```
In [71]: loss_value128 = history_dict_128["loss"]
val_loss_value128 = history_dict_128["val_loss"]
epochs_128 = range(1, len(loss_value128) + 1)
plt.plot(epochs_128, loss_value128, "bo", label="Training loss")
plt.plot(epochs_128, val_loss_value128, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()


plt.clf()
accuracy_128 = history_dict_128["accuracy"]
val_accuracy_128 = history_dict_128["val_accuracy"]
plt.plot(epochs_128, accuracy_128, "bo", label="Training acc")
plt.plot(epochs_128, val_accuracy_128, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```





In [72]:

```
history_128 = model_128units.fit(x_train, y_train, epochs=2, batch_size=512)
results_128_units = model_128units.evaluate(x_test, y_test)
results_128_units
```

```
Epoch 1/2
49/49 [==============================] - 1s 21ms/step - loss: 0.1788 - accuracy:
0.9447
Epoch 2/2
49/49 [==============================] - 1s 18ms/step - loss: 0.0821 - accuracy:
0.9736
782/782 [==============================] - 2s 3ms/step - loss: 0.4280 - accuracy:
0.8608
```

Out[72]:

```
[0.4279813766479492, 0.8608400225639343]
```

In [73]:

```
model_128units.predict(x_test)
```

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

Out[73]:

```
array([[0.00357643],
       [0.9999962 ],
       [0.0501473 ],
       ...,
       [0.00548558],
       [0.00419548],
       [0.83807635]], dtype=float32)
```

In [74]:

```
#MSE Loss Function model with 16 units and 3-layers
```

In [75]:

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

```
#Compilation of model
MSE_model.compile(optimizer="rmsprop",
              loss="mse",
              metrics=["accuracy"])

#Validation of model
x_val_MSE = x_train[:10000]
partial_x_train = x_train[10000:]

y_val_MSE = y_train[:10000]
partial_y_train = y_train[10000:]
# Model Fit

history_MSE = MSE_model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val_MSE, y_val_MSE))
```

```
Epoch 1/20
30/30 [==============================] - 1s 25ms/step - loss: 0.2123 - accuracy:
0.7470 - val_loss: 0.1711 - val_accuracy: 0.8009
Epoch 2/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1317 - accuracy:
0.8639 - val_loss: 0.1161 - val_accuracy: 0.8672
Epoch 3/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0930 - accuracy: 0.
8968 - val_loss: 0.0957 - val_accuracy: 0.8838
Epoch 4/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0737 - accuracy: 0.
9127 - val_loss: 0.0892 - val_accuracy: 0.8857
Epoch 5/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0618 - accuracy:
0.9279 - val_loss: 0.0848 - val_accuracy: 0.8866
Epoch 6/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0528 - accuracy: 0.
9401 - val_loss: 0.0931 - val_accuracy: 0.8718
Epoch 7/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0467 - accuracy: 0.
9469 - val_loss: 0.0962 - val_accuracy: 0.8679
Epoch 8/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0404 - accuracy:
0.9563 - val_loss: 0.0952 - val_accuracy: 0.8726
Epoch 9/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0364 - accuracy:
0.9617 - val_loss: 0.0874 - val_accuracy: 0.8789
Epoch 10/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0310 - accuracy:
0.9685 - val_loss: 0.0880 - val_accuracy: 0.8802
Epoch 11/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0293 - accuracy:
0.9715 - val_loss: 0.0886 - val_accuracy: 0.8801
Epoch 12/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0254 - accuracy: 0.
9756 - val_loss: 0.0938 - val_accuracy: 0.8744
Epoch 13/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0239 - accuracy:
0.9765 - val_loss: 0.0911 - val_accuracy: 0.8784
Epoch 14/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0202 - accuracy: 0.
9813 - val_loss: 0.0931 - val_accuracy: 0.8771
Epoch 15/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0208 - accuracy: 0.
9811 - val_loss: 0.1011 - val_accuracy: 0.8695
Epoch 16/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0165 - accuracy: 0.
9863 - val_loss: 0.0998 - val_accuracy: 0.8713
Epoch 17/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0155 - accuracy: 0.
9862 - val_loss: 0.1022 - val_accuracy: 0.8701
Epoch 18/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0151 - accuracy: 0.
9869 - val_loss: 0.0985 - val_accuracy: 0.8758
Epoch 19/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0145 - accuracy: 0.
9867 - val_loss: 0.0994 - val_accuracy: 0.8724
Epoch 20/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0122 - accuracy: 0.
9891 - val_loss: 0.1061 - val_accuracy: 0.8673
```

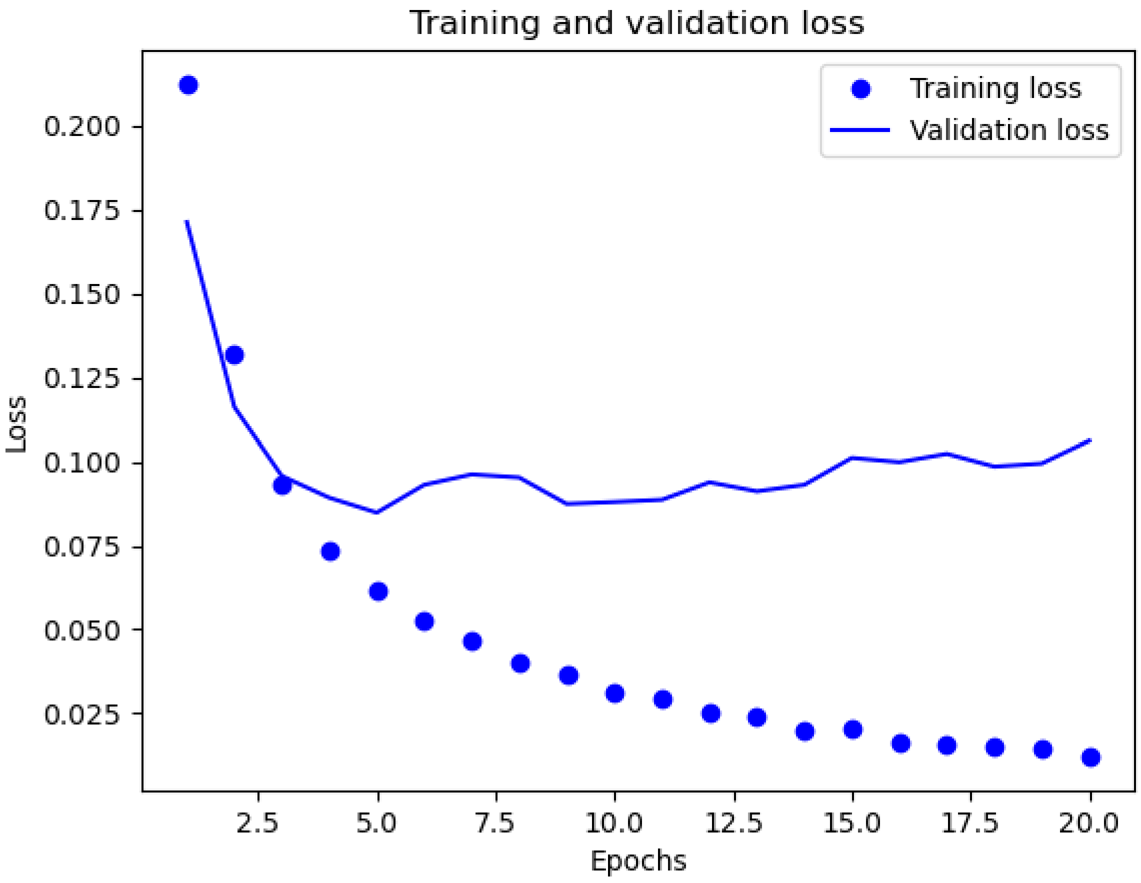
In [76]:

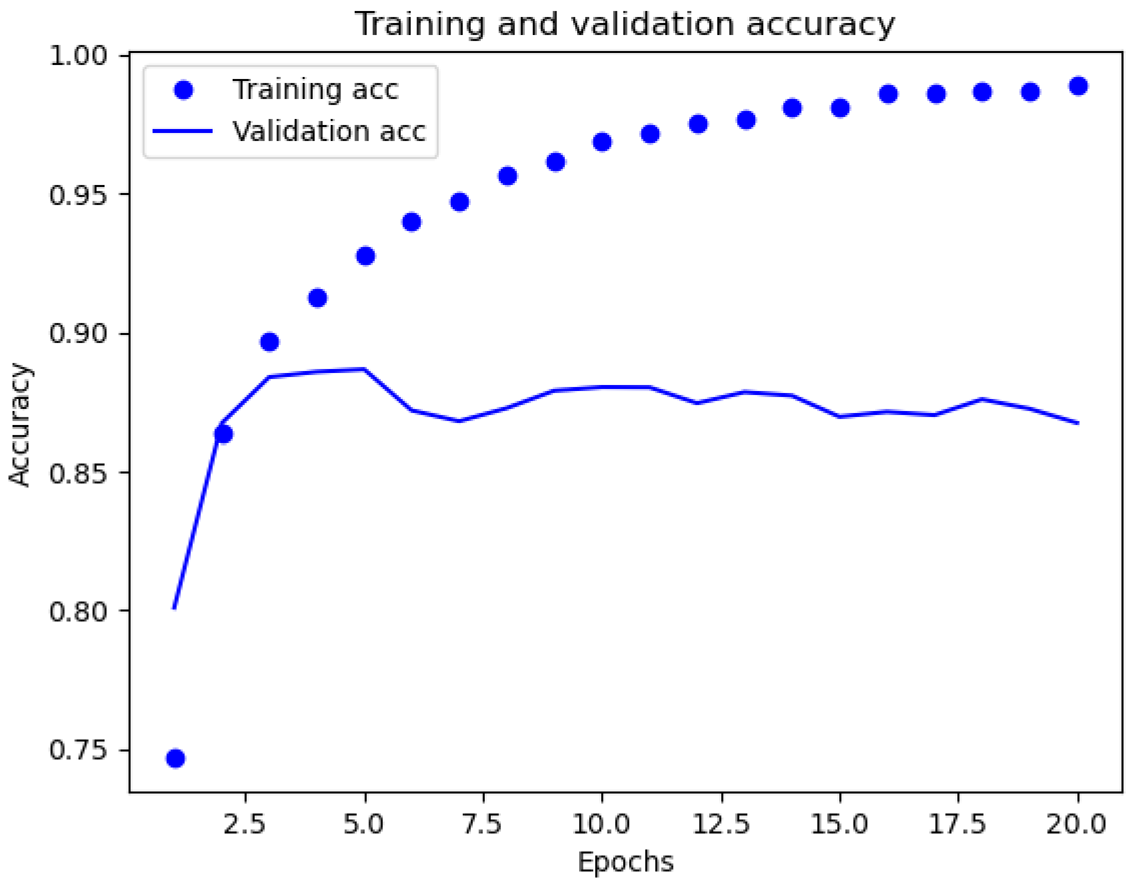
```
historydict_MSE = history_MSE.history
historydict_MSE.keys()
```

```
Out[76]: dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

```
In [77]: import matplotlib.pyplot as plt
loss_value_MSE = historydict_MSE["loss"]
val_loss_value_MSE = historydict_MSE["val_loss"]
epochs_MSE = range(1, len(loss_value_MSE) + 1)
plt.plot(epochs_MSE, loss_value_MSE, "bo", label="Training loss")
plt.plot(epochs_MSE, val_loss_value_MSE, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()


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





In [78]:
```
MSE_model.fit(x_train, y_train, epochs=8, batch_size=512)
results_MSE = MSE_model.evaluate(x_test, y_test)
results_MSE
```

```
Epoch 1/8
49/49 [==============================] - 0s 7ms/step - loss: 0.0484 - accuracy: 0.
9416
Epoch 2/8
49/49 [==============================] - 0s 7ms/step - loss: 0.0397 - accuracy: 0.
9550
Epoch 3/8
49/49 [==============================] - 0s 7ms/step - loss: 0.0352 - accuracy: 0.
9606
Epoch 4/8
49/49 [==============================] - 0s 7ms/step - loss: 0.0307 - accuracy: 0.
9666
Epoch 5/8
49/49 [==============================] - 0s 7ms/step - loss: 0.0301 - accuracy: 0.
9676
Epoch 6/8
49/49 [==============================] - 0s 7ms/step - loss: 0.0268 - accuracy: 0.
9722
Epoch 7/8
49/49 [==============================] - 0s 6ms/step - loss: 0.0254 - accuracy: 0.
9735
Epoch 8/8
49/49 [==============================] - 0s 6ms/step - loss: 0.0234 - accuracy: 0.
9766
782/782 [==============================] - 2s 2ms/step - loss: 0.1116 - accuracy:
0.8660
```

Out[78]:
```
[0.11159390956163406, 0.8660399913787842]
```

In [79]:
```
MSE_model.predict(x_test)
```

```
782/782 [==============================] - 1s 2ms/step
```

Out[79]:

```
array([[0.00609913],
       [0.99997556],
       [0.90007144],
       ...,
       [0.03745234],
       [0.01846988],
       [0.764864  ]], dtype=float32)
```

In [80]:

```
#tanh activation

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

tanh.compile(optimizer='rmsprop',
              loss='mse',
              metrics=['accuracy'])

x_val_tanh = x_train[:10000]
partial_x_train = x_train[10000:]

y_val_tanh = y_train[:10000]
partial_y_train = y_train[10000:]


historytanh_model = tanh.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val_tanh, y_val_tanh))
```

```
Epoch 1/20
30/30 [==============================] - 1s 26ms/step - loss: 0.1775 - accuracy:
0.7787 - val_loss: 0.1338 - val_accuracy: 0.8625
Epoch 2/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1109 - accuracy:
0.8893 - val_loss: 0.1179 - val_accuracy: 0.8498
Epoch 3/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0877 - accuracy:
0.9074 - val_loss: 0.0996 - val_accuracy: 0.8748
Epoch 4/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0744 - accuracy:
0.9202 - val_loss: 0.0903 - val_accuracy: 0.8883
Epoch 5/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0645 - accuracy: 0.
9316 - val_loss: 0.0863 - val_accuracy: 0.8887
Epoch 6/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0579 - accuracy: 0.
9379 - val_loss: 0.0848 - val_accuracy: 0.8876
Epoch 7/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0520 - accuracy:
0.9455 - val_loss: 0.0856 - val_accuracy: 0.8859
Epoch 8/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0471 - accuracy: 0.
9519 - val_loss: 0.0833 - val_accuracy: 0.8847
Epoch 9/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0435 - accuracy:
0.9568 - val_loss: 0.0860 - val_accuracy: 0.8840
Epoch 10/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0397 - accuracy:
0.9606 - val_loss: 0.0836 - val_accuracy: 0.8839
Epoch 11/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0364 - accuracy: 0.
9651 - val_loss: 0.0843 - val_accuracy: 0.8837
Epoch 12/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0338 - accuracy:
0.9689 - val_loss: 0.0874 - val_accuracy: 0.8810
Epoch 13/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0312 - accuracy:
0.9711 - val_loss: 0.0875 - val_accuracy: 0.8777
Epoch 14/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0289 - accuracy:
0.9737 - val_loss: 0.0868 - val_accuracy: 0.8817
Epoch 15/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0263 - accuracy: 0.
9773 - val_loss: 0.0950 - val_accuracy: 0.8747
Epoch 16/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0241 - accuracy: 0.
9795 - val_loss: 0.0887 - val_accuracy: 0.8795
Epoch 17/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0232 - accuracy:
0.9811 - val_loss: 0.0915 - val_accuracy: 0.8772
Epoch 18/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0213 - accuracy:
0.9837 - val_loss: 0.0942 - val_accuracy: 0.8719
Epoch 19/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0199 - accuracy:
0.9847 - val_loss: 0.0919 - val_accuracy: 0.8766
Epoch 20/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0186 - accuracy: 0.
9863 - val_loss: 0.0929 - val_accuracy: 0.8767
```

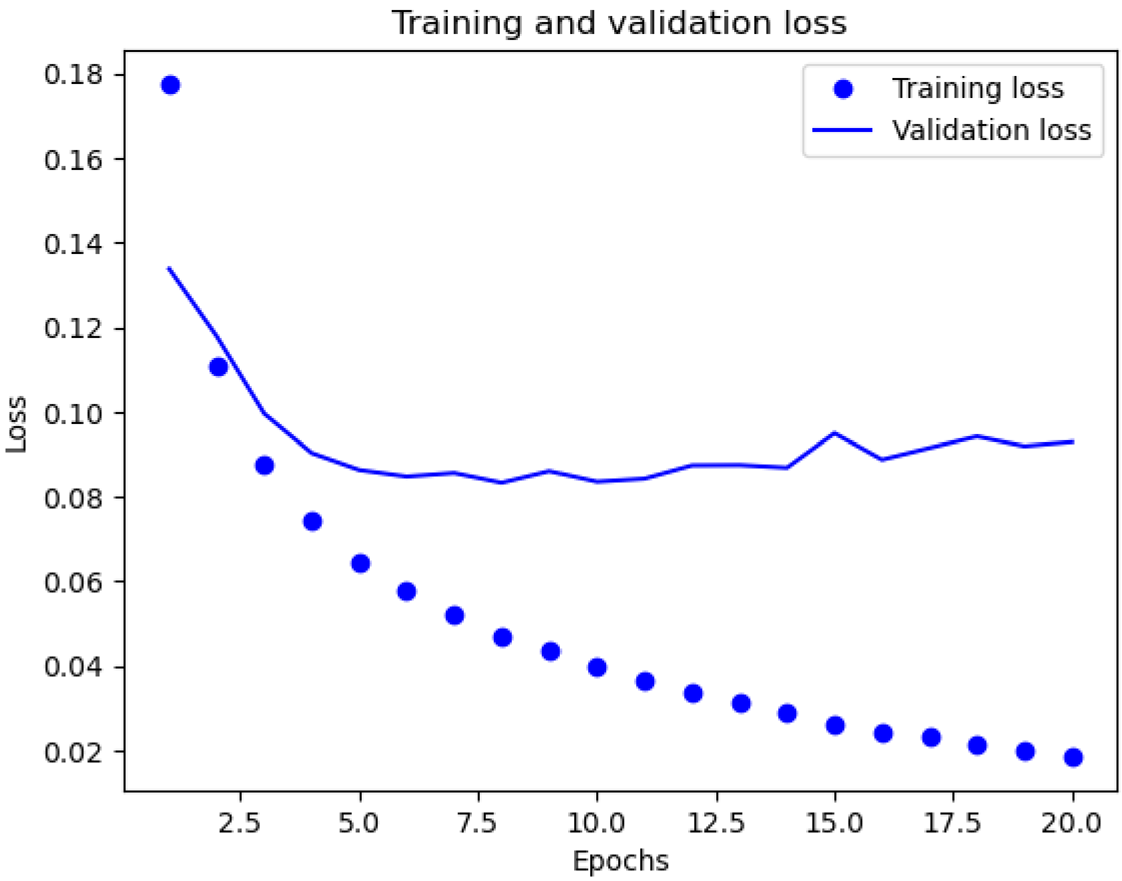
In [81]:

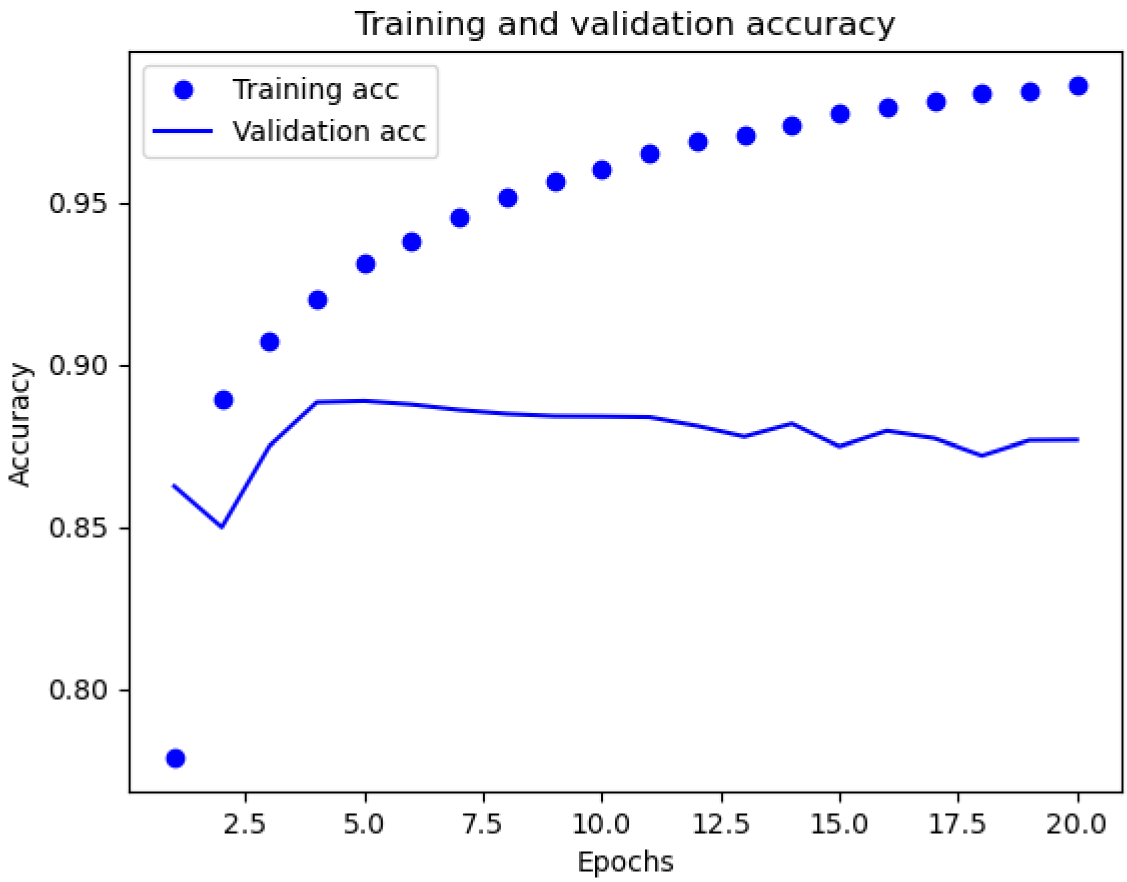
```
historydict_tanh = historytanh_model.history
historydict_tanh.keys()
```

Out[81]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [82]:
```
loss_value_tanh= historydict_tanh["loss"]
val_loss_value_tanh = historydict_tanh["val_loss"]
epochs_tanh = range(1, len(loss_value_tanh) + 1)
plt.plot(epochs_tanh, loss_value_tanh, "bo", label="Training loss")
plt.plot(epochs_tanh, val_loss_value_tanh, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()

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





In [83]:
```
tanh.fit(x_train, y_train, epochs=8, batch_size=512)
results_tanh = tanh.evaluate(x_test, y_test)
results_tanh
```

```
Epoch 1/8
49/49 [==============================] - 0s 7ms/step - loss: 0.0476 - accuracy: 0.
9421
Epoch 2/8
49/49 [==============================] - 0s 7ms/step - loss: 0.0409 - accuracy: 0.
9526
Epoch 3/8
49/49 [==============================] - 0s 6ms/step - loss: 0.0369 - accuracy: 0.
9585
Epoch 4/8
49/49 [==============================] - 0s 7ms/step - loss: 0.0343 - accuracy: 0.
9638
Epoch 5/8
49/49 [==============================] - 0s 6ms/step - loss: 0.0318 - accuracy: 0.
9666
Epoch 6/8
49/49 [==============================] - 0s 7ms/step - loss: 0.0298 - accuracy: 0.
9690
Epoch 7/8
49/49 [==============================] - 0s 6ms/step - loss: 0.0275 - accuracy: 0.
9722
Epoch 8/8
49/49 [==============================] - 0s 6ms/step - loss: 0.0262 - accuracy: 0.
9742
782/782 [==============================] - 1s 2ms/step - loss: 0.1040 - accuracy:
0.8690
```

Out[83]:
```
[0.10404925048351288, 0.8690000176429749]
```

In [84]:
```
#Adam Operator with 16 units and 3 layers
```

In [85]:

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

adam.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
x_adam = x_train[:10000]
partial_x_train = x_train[10000:]

y_adam = y_train[:10000]
partial_y_train = y_train[10000:]

historyadam = adam.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_adam, y_adam))
```

```
Epoch 1/20
30/30 [==============================] - 2s 28ms/step - loss: 0.6091 - accuracy:
0.5687 - val_loss: 0.5244 - val_accuracy: 0.7669
Epoch 2/20
30/30 [==============================] - 0s 10ms/step - loss: 0.4496 - accuracy:
0.8680 - val_loss: 0.4395 - val_accuracy: 0.8729
Epoch 3/20
30/30 [==============================] - 0s 10ms/step - loss: 0.3171 - accuracy:
0.9345 - val_loss: 0.3451 - val_accuracy: 0.8812
Epoch 4/20
30/30 [==============================] - 0s 11ms/step - loss: 0.1740 - accuracy:
0.9588 - val_loss: 0.3138 - val_accuracy: 0.8823
Epoch 5/20
30/30 [==============================] - 0s 12ms/step - loss: 0.1038 - accuracy:
0.9769 - val_loss: 0.3453 - val_accuracy: 0.8769
Epoch 6/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0682 - accuracy:
0.9871 - val_loss: 0.4098 - val_accuracy: 0.8730
Epoch 7/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0467 - accuracy:
0.9926 - val_loss: 0.4705 - val_accuracy: 0.8693
Epoch 8/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0352 - accuracy:
0.9954 - val_loss: 0.4994 - val_accuracy: 0.8708
Epoch 9/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0284 - accuracy:
0.9959 - val_loss: 0.5421 - val_accuracy: 0.8694
Epoch 10/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0249 - accuracy:
0.9961 - val_loss: 0.5660 - val_accuracy: 0.8692
Epoch 11/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0225 - accuracy:
0.9963 - val_loss: 0.5892 - val_accuracy: 0.8681
Epoch 12/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0201 - accuracy:
0.9965 - val_loss: 0.6065 - val_accuracy: 0.8686
Epoch 13/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0179 - accuracy: 0.
9967 - val_loss: 0.6351 - val_accuracy: 0.8676
Epoch 14/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0152 - accuracy:
0.9969 - val_loss: 0.6726 - val_accuracy: 0.8657
Epoch 15/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0123 - accuracy: 0.
9972 - val_loss: 0.6563 - val_accuracy: 0.8678
Epoch 16/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0092 - accuracy:
0.9978 - val_loss: 0.7062 - val_accuracy: 0.8666
Epoch 17/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0065 - accuracy:
0.9984 - val_loss: 0.7216 - val_accuracy: 0.8679
Epoch 18/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0047 - accuracy:
0.9988 - val_loss: 0.7631 - val_accuracy: 0.8672
Epoch 19/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0036 - accuracy: 0.
9990 - val_loss: 0.7633 - val_accuracy: 0.8685
Epoch 20/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0027 - accuracy: 0.
9991 - val_loss: 0.7914 - val_accuracy: 0.8674
```

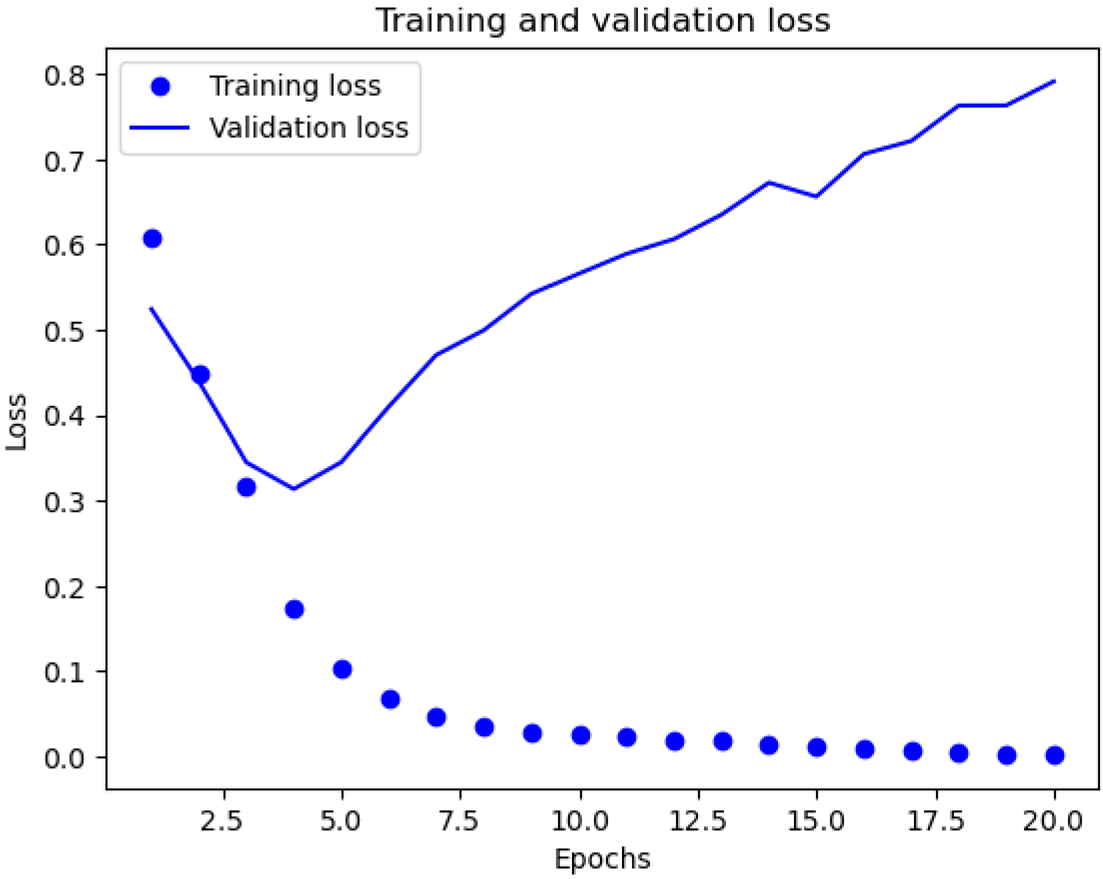
In [86]:

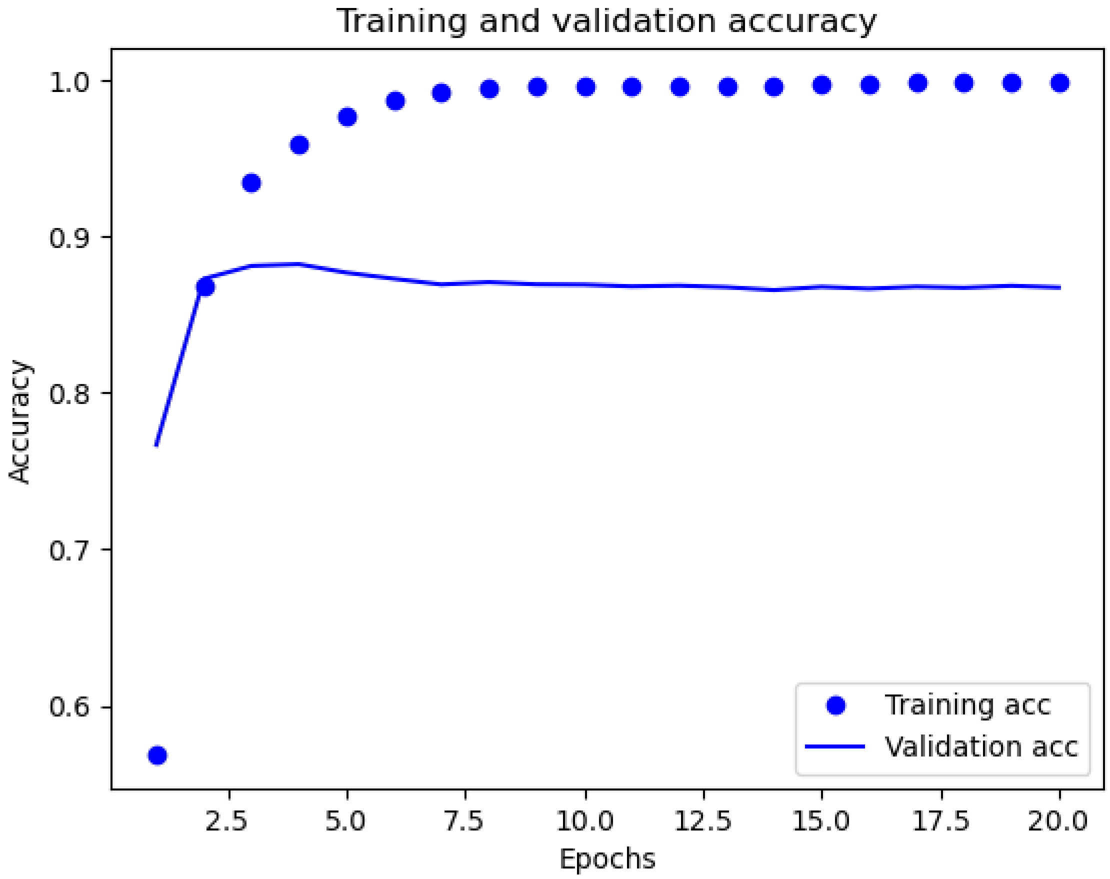
```
historydict_adam = historyadam.history
historydict_adam.keys()
```

```
Out[86]: dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

```
In [87]: loss_value_adam = historydict_adam["loss"]
         val_loss_value_adam = historydict_adam["val_loss"]
         epochs_adam = range(1, len(loss_value_adam) + 1)
         plt.plot(epochs_adam, loss_value_adam, "bo", label="Training loss")
         plt.plot(epochs_adam, val_loss_value_adam, "b", label="Validation loss")
         plt.title("Training and validation loss")
         plt.xlabel("Epochs")
         plt.ylabel("Loss")
         plt.legend()
         plt.show()


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





In [88]:
```
adam.fit(x_train, y_train, epochs=4, batch_size=512)
results_adam = adam.evaluate(x_test, y_test)
results_adam
```

```
Epoch 1/4
49/49 [==============================] - 0s 8ms/step - loss: 0.2444 - accuracy: 0.
9266
Epoch 2/4
49/49 [==============================] - 0s 7ms/step - loss: 0.1059 - accuracy: 0.
9622
Epoch 3/4
49/49 [==============================] - 0s 7ms/step - loss: 0.0526 - accuracy: 0.
9847
Epoch 4/4
49/49 [==============================] - 0s 7ms/step - loss: 0.0287 - accuracy: 0.
9934
782/782 [==============================] - 1s 1ms/step - loss: 0.5853 - accuracy:
0.8588
```

Out[88]:
```
[0.5852583050727844, 0.8587599992752075]
```

In [89]:
```
#Regularization with 16 units and 2-layers
```

In [90]:
```
from tensorflow.keras import regularizers
regularization = keras.Sequential([
    layers.Dense(16, activation="relu",kernel_regularizer=regularizers.l2(0.001)),
    layers.Dense(16, activation="relu",kernel_regularizer=regularizers.l2(0.001)),
    layers.Dense(1, activation="sigmoid")
])
regularization.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])

history_regularization = regularization.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
historydict_regularization = history_regularization.history
historydict_regularization.keys()
```

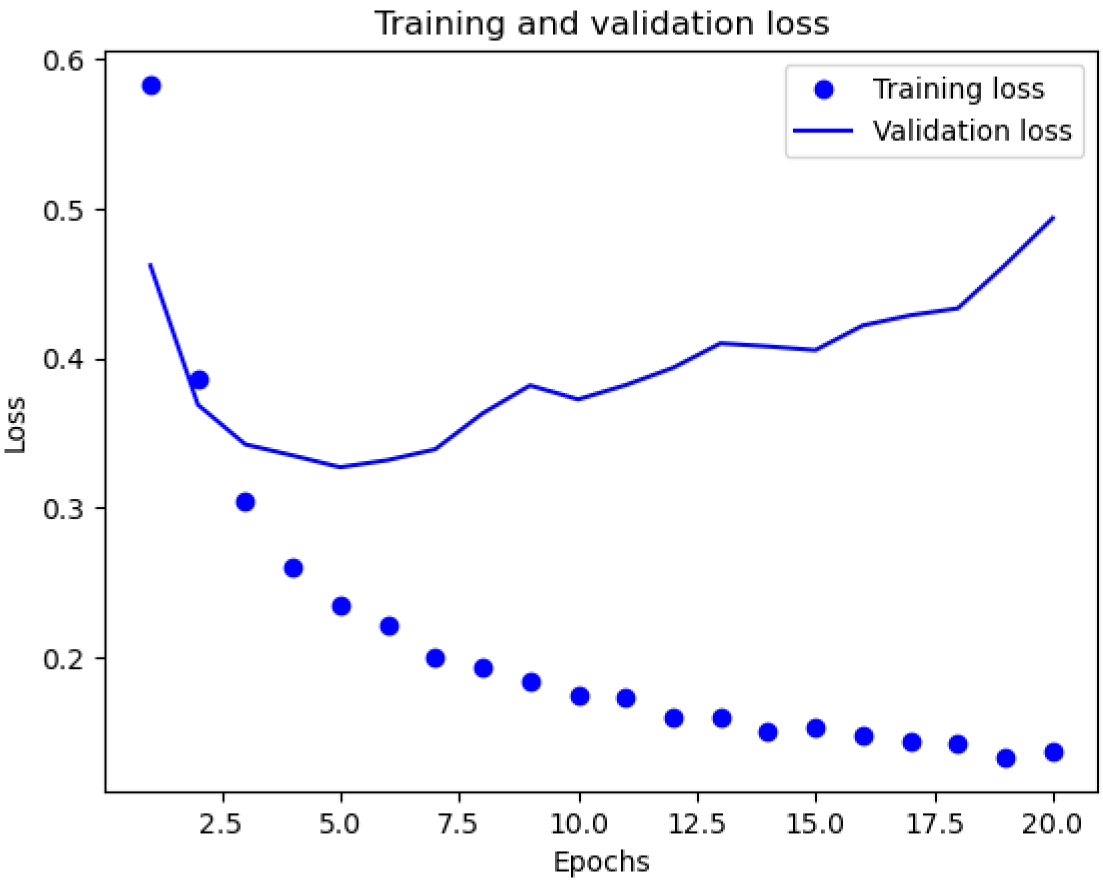
```
Epoch 1/20
30/30 [==============================] - 1s 27ms/step - loss: 0.5831 - accuracy:
0.7757 - val_loss: 0.4621 - val_accuracy: 0.8595
Epoch 2/20
30/30 [==============================] - 0s 10ms/step - loss: 0.3863 - accuracy:
0.8898 - val_loss: 0.3689 - val_accuracy: 0.8827
Epoch 3/20
30/30 [==============================] - 0s 10ms/step - loss: 0.3047 - accuracy:
0.9153 - val_loss: 0.3423 - val_accuracy: 0.8855
Epoch 4/20
30/30 [==============================] - 0s 10ms/step - loss: 0.2596 - accuracy:
0.9297 - val_loss: 0.3349 - val_accuracy: 0.8851
Epoch 5/20
30/30 [==============================] - 0s 10ms/step - loss: 0.2346 - accuracy:
0.9373 - val_loss: 0.3270 - val_accuracy: 0.8874
Epoch 6/20
30/30 [==============================] - 0s 10ms/step - loss: 0.2212 - accuracy:
0.9445 - val_loss: 0.3318 - val_accuracy: 0.8851
Epoch 7/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1995 - accuracy:
0.9521 - val_loss: 0.3390 - val_accuracy: 0.8853
Epoch 8/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1937 - accuracy:
0.9556 - val_loss: 0.3631 - val_accuracy: 0.8773
Epoch 9/20
30/30 [==============================] - 0s 9ms/step - loss: 0.1833 - accuracy: 0.
9604 - val_loss: 0.3818 - val_accuracy: 0.8730
Epoch 10/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1748 - accuracy:
0.9643 - val_loss: 0.3724 - val_accuracy: 0.8812
Epoch 11/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1728 - accuracy:
0.9645 - val_loss: 0.3820 - val_accuracy: 0.8797
Epoch 12/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1601 - accuracy:
0.9701 - val_loss: 0.3936 - val_accuracy: 0.8745
Epoch 13/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1595 - accuracy:
0.9699 - val_loss: 0.4099 - val_accuracy: 0.8749
Epoch 14/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1510 - accuracy:
0.9750 - val_loss: 0.4079 - val_accuracy: 0.8745
Epoch 15/20
30/30 [==============================] - 0s 9ms/step - loss: 0.1528 - accuracy: 0.
9734 - val_loss: 0.4054 - val_accuracy: 0.8772
Epoch 16/20
30/30 [==============================] - 0s 9ms/step - loss: 0.1473 - accuracy: 0.
9743 - val_loss: 0.4219 - val_accuracy: 0.8712
Epoch 17/20
30/30 [==============================] - 0s 9ms/step - loss: 0.1436 - accuracy: 0.
9767 - val_loss: 0.4287 - val_accuracy: 0.8743
Epoch 18/20
30/30 [==============================] - 0s 9ms/step - loss: 0.1426 - accuracy: 0.
9764 - val_loss: 0.4333 - val_accuracy: 0.8730
Epoch 19/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1330 - accuracy:
0.9815 - val_loss: 0.4624 - val_accuracy: 0.8643
Epoch 20/20
30/30 [==============================] - 0s 8ms/step - loss: 0.1364 - accuracy: 0.
9793 - val_loss: 0.4937 - val_accuracy: 0.8577
```

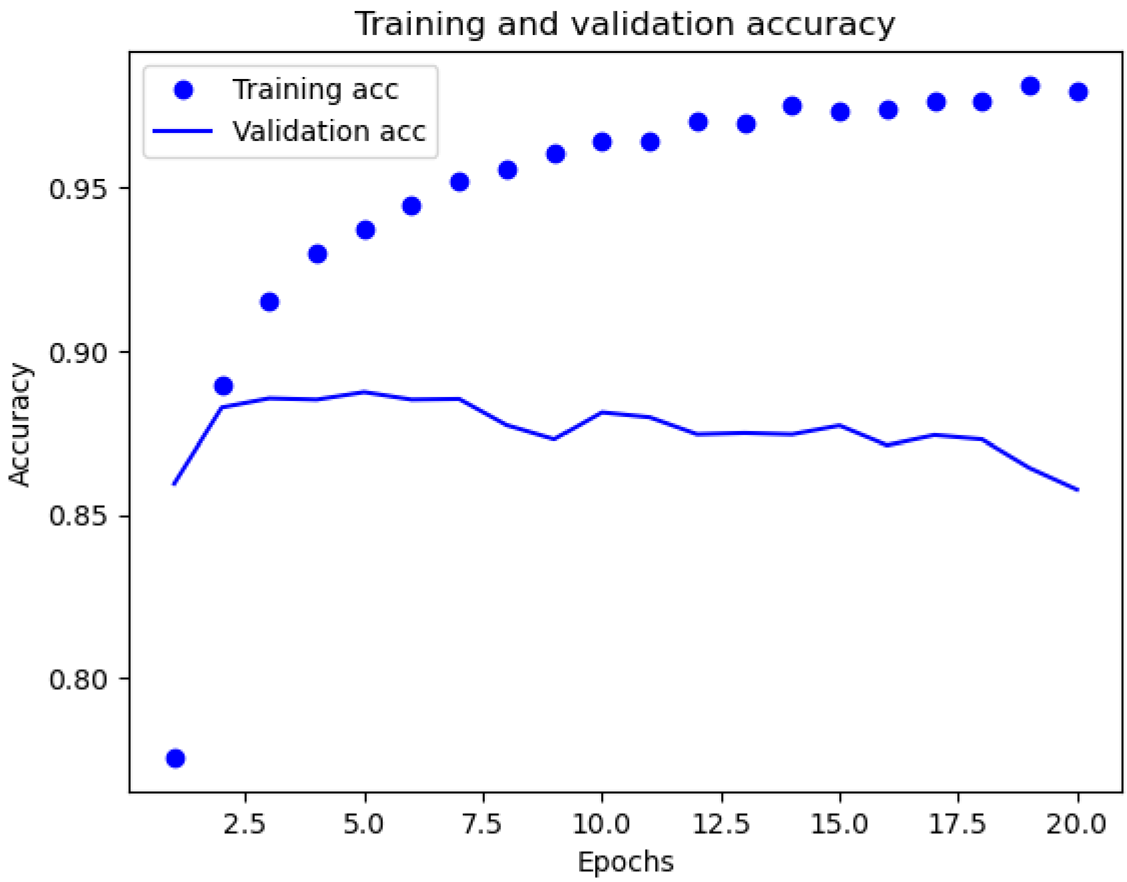
Out[90]:

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

In [91]:
```
loss_valu = historydict_regularization["loss"]
val_loss_value_r = historydict_regularization["val_loss"]
epochs_r = range(1, len(loss_valu) + 1)
plt.plot(epochs_r, loss_valu, "bo", label="Training loss")
plt.plot(epochs_r, val_loss_value_r, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()

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





In [92]:

```
regularization.fit(x_train, y_train, epochs=8, batch_size=512)
results_regularization = regularization.evaluate(x_test, y_test)
results_regularization
```

```
Epoch 1/8
49/49 [==============================] - 0s 7ms/step - loss: 0.2514 - accuracy: 0.
9356
Epoch 2/8
49/49 [==============================] - 0s 7ms/step - loss: 0.2090 - accuracy: 0.
9466
Epoch 3/8
49/49 [==============================] - 0s 7ms/step - loss: 0.1980 - accuracy: 0.
9512
Epoch 4/8
49/49 [==============================] - 0s 7ms/step - loss: 0.1859 - accuracy: 0.
9552
Epoch 5/8
49/49 [==============================] - 0s 7ms/step - loss: 0.1798 - accuracy: 0.
9580
Epoch 6/8
49/49 [==============================] - 0s 6ms/step - loss: 0.1745 - accuracy: 0.
9588
Epoch 7/8
49/49 [==============================] - 0s 6ms/step - loss: 0.1676 - accuracy: 0.
9633
Epoch 8/8
49/49 [==============================] - 0s 6ms/step - loss: 0.1670 - accuracy: 0.
9632
782/782 [==============================] - 1s 2ms/step - loss: 0.4251 - accuracy:
0.8691
```

Out[92]:

```
[0.4250975251197815, 0.8691200017929077]
```

In [93]:

```
#Dropout function with 16 units and 3-layers
```

In [94]:

```
from tensorflow.keras import regularizers
Dropout = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dropout(0.5),
    layers.Dense(16, activation="relu"),
    layers.Dropout(0.5),
    layers.Dense(16, activation="relu"),
    layers.Dropout(0.5),
    layers.Dense(1, activation="sigmoid")
])
Dropout.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])

history_Dropout = Dropout.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
historydict_Dropout = history_Dropout.history
historydict_Dropout.keys()
```

```
Epoch 1/20
30/30 [==============================] - 2s 28ms/step - loss: 0.6594 - accuracy:
0.5935 - val_loss: 0.5658 - val_accuracy: 0.8151
Epoch 2/20
30/30 [==============================] - 0s 11ms/step - loss: 0.5669 - accuracy:
0.7048 - val_loss: 0.4641 - val_accuracy: 0.8488
Epoch 3/20
30/30 [==============================] - 0s 10ms/step - loss: 0.4921 - accuracy:
0.7680 - val_loss: 0.3769 - val_accuracy: 0.8730
Epoch 4/20
30/30 [==============================] - 0s 11ms/step - loss: 0.4216 - accuracy:
0.8199 - val_loss: 0.3249 - val_accuracy: 0.8810
Epoch 5/20
30/30 [==============================] - 0s 10ms/step - loss: 0.3768 - accuracy:
0.8489 - val_loss: 0.2927 - val_accuracy: 0.8813
Epoch 6/20
30/30 [==============================] - 0s 12ms/step - loss: 0.3248 - accuracy:
0.8777 - val_loss: 0.2870 - val_accuracy: 0.8855
Epoch 7/20
30/30 [==============================] - 0s 10ms/step - loss: 0.2916 - accuracy:
0.8931 - val_loss: 0.2925 - val_accuracy: 0.8856
Epoch 8/20
30/30 [==============================] - 0s 11ms/step - loss: 0.2646 - accuracy:
0.9043 - val_loss: 0.2995 - val_accuracy: 0.8864
Epoch 9/20
30/30 [==============================] - 0s 11ms/step - loss: 0.2447 - accuracy:
0.9121 - val_loss: 0.3348 - val_accuracy: 0.8865
Epoch 10/20
30/30 [==============================] - 0s 11ms/step - loss: 0.2261 - accuracy:
0.9187 - val_loss: 0.3305 - val_accuracy: 0.8840
Epoch 11/20
30/30 [==============================] - 0s 10ms/step - loss: 0.2091 - accuracy:
0.9250 - val_loss: 0.3517 - val_accuracy: 0.8826
Epoch 12/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1903 - accuracy:
0.9306 - val_loss: 0.4040 - val_accuracy: 0.8868
Epoch 13/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1843 - accuracy:
0.9319 - val_loss: 0.4021 - val_accuracy: 0.8811
Epoch 14/20
30/30 [==============================] - 0s 11ms/step - loss: 0.1751 - accuracy:
0.9373 - val_loss: 0.4359 - val_accuracy: 0.8835
Epoch 15/20
30/30 [==============================] - 0s 11ms/step - loss: 0.1684 - accuracy:
0.9393 - val_loss: 0.4774 - val_accuracy: 0.8812
Epoch 16/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1593 - accuracy:
0.9407 - val_loss: 0.4763 - val_accuracy: 0.8805
Epoch 17/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1503 - accuracy:
0.9463 - val_loss: 0.5285 - val_accuracy: 0.8839
Epoch 18/20
30/30 [==============================] - 0s 13ms/step - loss: 0.1438 - accuracy:
0.9473 - val_loss: 0.5491 - val_accuracy: 0.8834
Epoch 19/20
30/30 [==============================] - 0s 11ms/step - loss: 0.1435 - accuracy:
0.9504 - val_loss: 0.5622 - val_accuracy: 0.8807
Epoch 20/20
30/30 [==============================] - 0s 12ms/step - loss: 0.1442 - accuracy:
0.9481 - val_loss: 0.5709 - val_accuracy: 0.8790
```

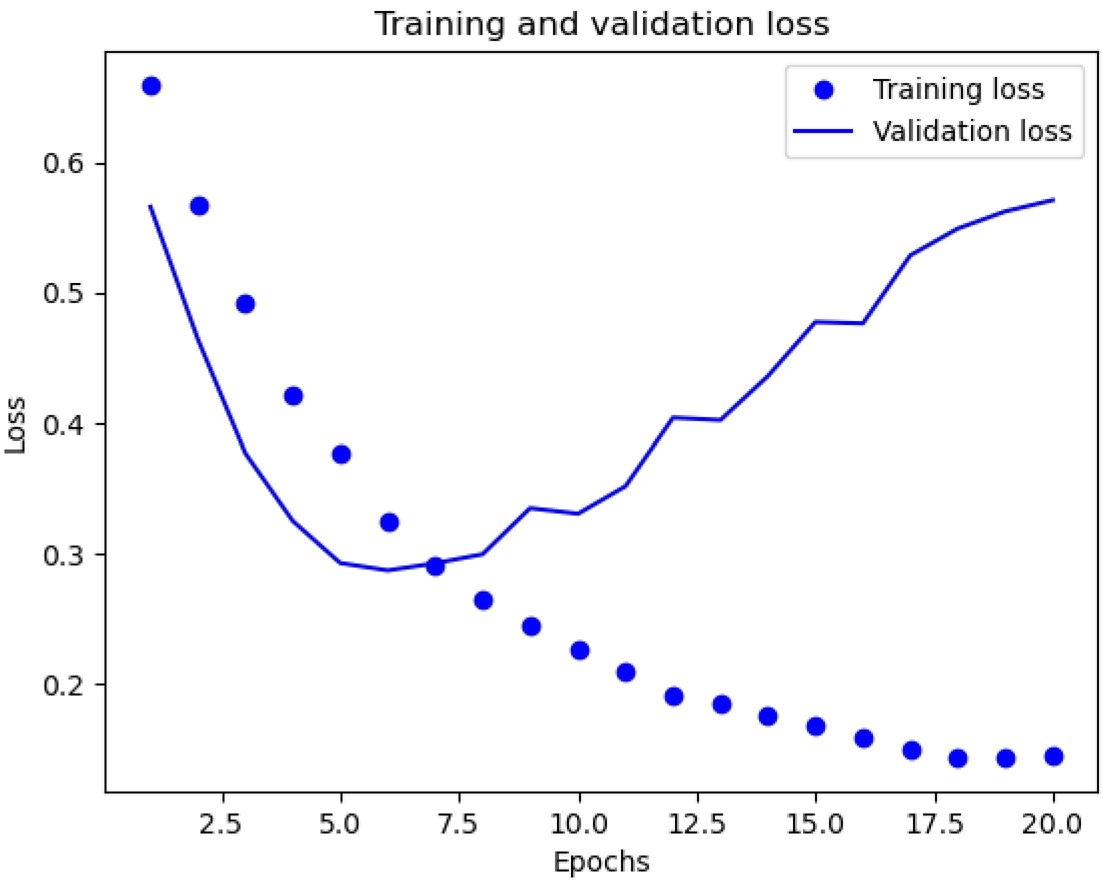
Out[94]:

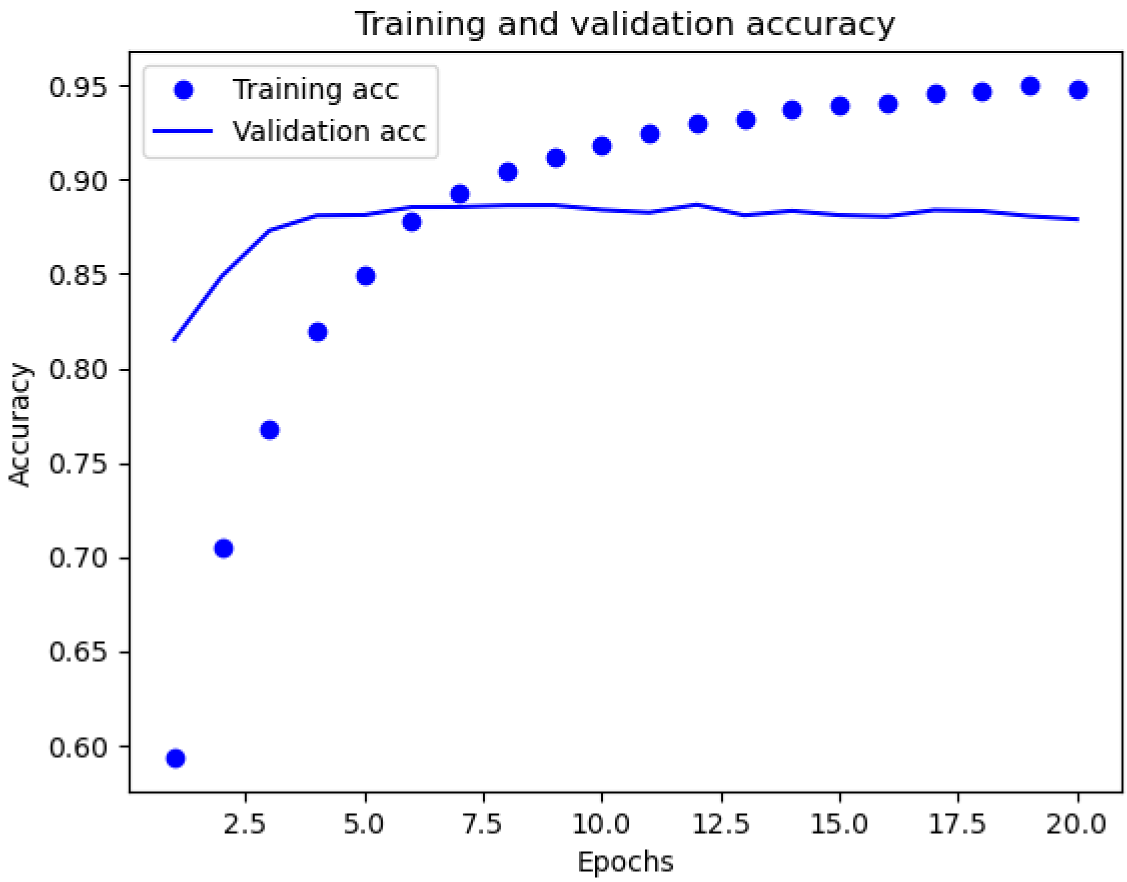
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [95]:

```
loss_val = historydict_Dropout["loss"]
val_loss_val_d = historydict_Dropout["val_loss"]
epochs_d = range(1, len(loss_val) + 1)
plt.plot(epochs_d, loss_val, "bo", label="Training loss")
plt.plot(epochs_d, val_loss_val_d, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()


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





In [96]:
```
Dropout.fit(x_train, y_train, epochs=8, batch_size=512)
results_Dropout = Dropout.evaluate(x_test, y_test)
results_Dropout
```

```
Epoch 1/8
49/49 [==============================] - 1s 10ms/step - loss: 0.3214 - accuracy:
0.8984
Epoch 2/8
49/49 [==============================] - 0s 9ms/step - loss: 0.2763 - accuracy: 0.
9095
Epoch 3/8
49/49 [==============================] - 0s 9ms/step - loss: 0.2513 - accuracy: 0.
9159
Epoch 4/8
49/49 [==============================] - 0s 9ms/step - loss: 0.2224 - accuracy: 0.
9233
Epoch 5/8
49/49 [==============================] - 0s 9ms/step - loss: 0.2110 - accuracy: 0.
9272
Epoch 6/8
49/49 [==============================] - 0s 8ms/step - loss: 0.2017 - accuracy: 0.
9306
Epoch 7/8
49/49 [==============================] - 0s 8ms/step - loss: 0.1966 - accuracy: 0.
9324
Epoch 8/8
49/49 [==============================] - 0s 8ms/step - loss: 0.1867 - accuracy: 0.
9360
782/782 [==============================] - 2s 2ms/step - loss: 0.4848 - accuracy:
0.8762
```

Out[96]:
```
[0.48477423191070557, 0.8762400150299072]
```

In [97]:
```
#Training model with hyper tuned parameters with 32 units and 3 -layers
#Training model with hyper tuned parameters
```

In [98]:

```
from tensorflow.keras import regularizers
Hyper = keras.Sequential([
    layers.Dense(32, activation="relu",kernel_regularizer=regularizers.l2(0.0001)),
    layers.Dropout(0.5),
    layers.Dense(32, activation="relu",kernel_regularizer=regularizers.l2(0.0001)),
    layers.Dropout(0.5),
    layers.Dense(16, activation="relu",kernel_regularizer=regularizers.l2(0.0001)),
    layers.Dropout(0.5),
    layers.Dense(1, activation="sigmoid")
])
Hyper.compile(optimizer="rmsprop",
              loss="mse",
              metrics=["accuracy"])

history_Hyper = Hyper.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
history_dictHyper = history_Hyper.history
history_dictHyper.keys()
```

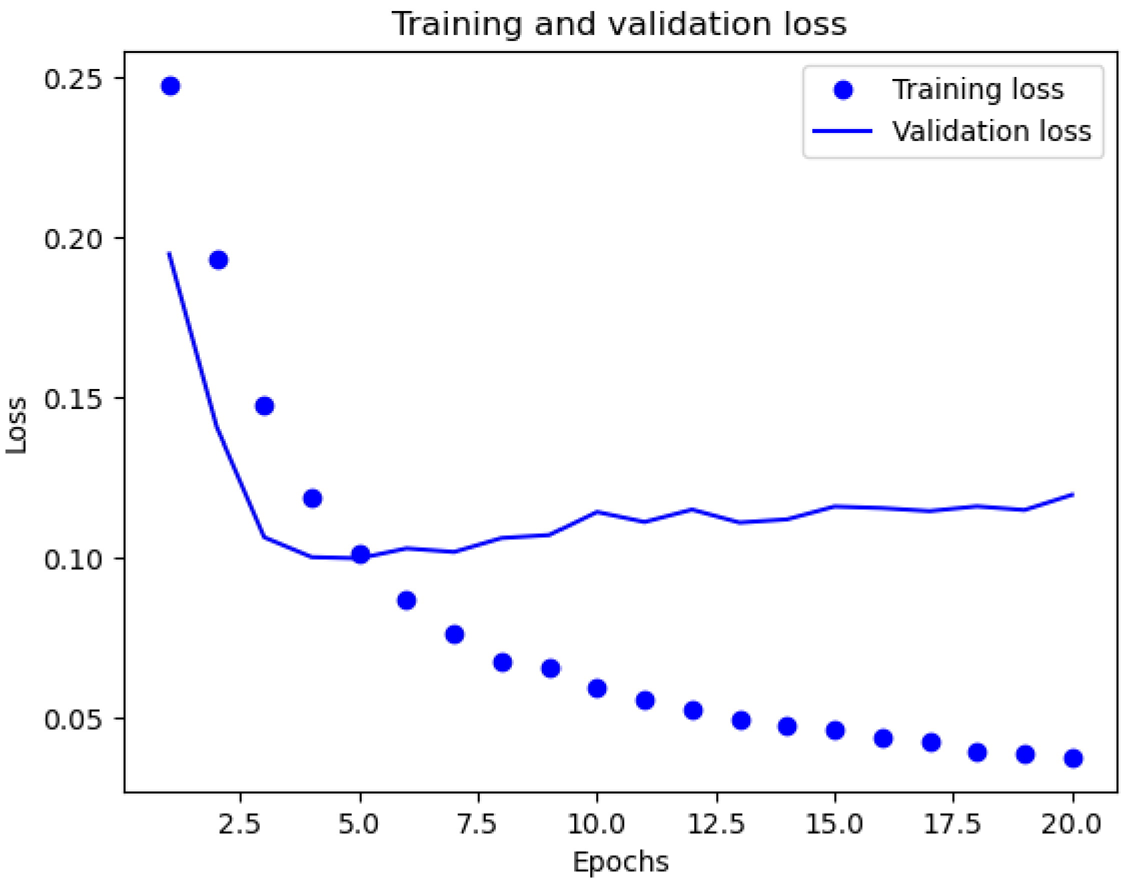
```
Epoch 1/20
30/30 [==============================] - 2s 34ms/step - loss: 0.2476 - accuracy:
0.5794 - val_loss: 0.1947 - val_accuracy: 0.8217
Epoch 2/20
30/30 [==============================] - 0s 14ms/step - loss: 0.1929 - accuracy:
0.7506 - val_loss: 0.1408 - val_accuracy: 0.8416
Epoch 3/20
30/30 [==============================] - 0s 13ms/step - loss: 0.1473 - accuracy:
0.8271 - val_loss: 0.1064 - val_accuracy: 0.8773
Epoch 4/20
30/30 [==============================] - 0s 14ms/step - loss: 0.1191 - accuracy:
0.8706 - val_loss: 0.1002 - val_accuracy: 0.8803
Epoch 5/20
30/30 [==============================] - 0s 14ms/step - loss: 0.1015 - accuracy:
0.8898 - val_loss: 0.0998 - val_accuracy: 0.8834
Epoch 6/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0870 - accuracy:
0.9117 - val_loss: 0.1029 - val_accuracy: 0.8834
Epoch 7/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0766 - accuracy:
0.9250 - val_loss: 0.1018 - val_accuracy: 0.8858
Epoch 8/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0678 - accuracy:
0.9352 - val_loss: 0.1062 - val_accuracy: 0.8819
Epoch 9/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0659 - accuracy:
0.9388 - val_loss: 0.1071 - val_accuracy: 0.8812
Epoch 10/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0597 - accuracy:
0.9463 - val_loss: 0.1143 - val_accuracy: 0.8781
Epoch 11/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0560 - accuracy:
0.9510 - val_loss: 0.1112 - val_accuracy: 0.8807
Epoch 12/20
30/30 [==============================] - 0s 15ms/step - loss: 0.0524 - accuracy:
0.9561 - val_loss: 0.1150 - val_accuracy: 0.8765
Epoch 13/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0495 - accuracy:
0.9585 - val_loss: 0.1110 - val_accuracy: 0.8823
Epoch 14/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0475 - accuracy:
0.9615 - val_loss: 0.1120 - val_accuracy: 0.8807
Epoch 15/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0462 - accuracy:
0.9631 - val_loss: 0.1160 - val_accuracy: 0.8787
Epoch 16/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0441 - accuracy:
0.9656 - val_loss: 0.1155 - val_accuracy: 0.8790
Epoch 17/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0427 - accuracy:
0.9661 - val_loss: 0.1146 - val_accuracy: 0.8783
Epoch 18/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0396 - accuracy:
0.9704 - val_loss: 0.1161 - val_accuracy: 0.8799
Epoch 19/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0388 - accuracy:
0.9702 - val_loss: 0.1149 - val_accuracy: 0.8822
Epoch 20/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0377 - accuracy:
0.9725 - val_loss: 0.1197 - val_accuracy: 0.8777
```

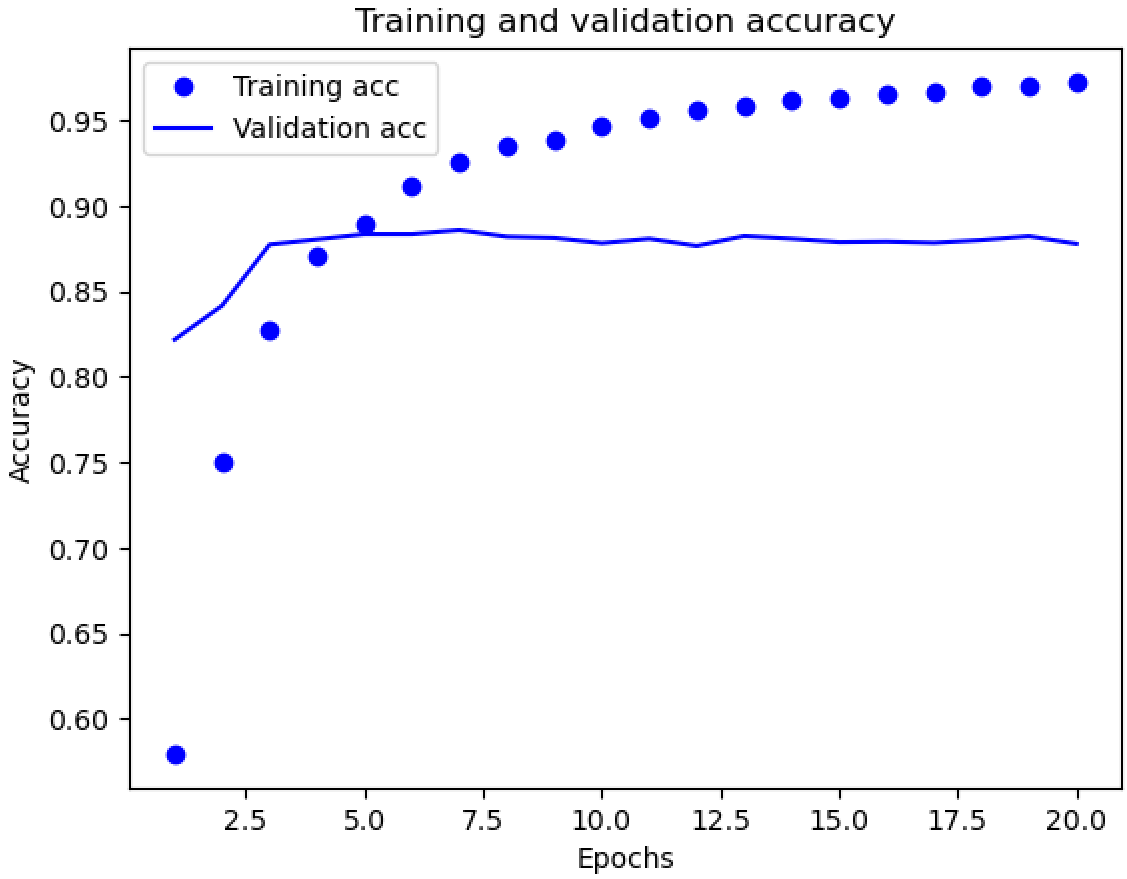
```
Out[98]: dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [99]:

```
loss_va_h = history_dictHyper["loss"]
val_loss_va_h = history_dictHyper["val_loss"]
epochs_h = range(1, len(loss_va_h) + 1)
plt.plot(epochs_h, loss_va_h, "bo", label="Training loss")
plt.plot(epochs_h, val_loss_va_h, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()


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





In [100…

```
Hyper.fit(x_train, y_train, epochs=8, batch_size=512)
results_Hyper = Hyper.evaluate(x_test, y_test)
results_Hyper
```

```
Epoch 1/8
49/49 [==============================] - 1s 10ms/step - loss: 0.0719 - accuracy:
0.9302
Epoch 2/8
49/49 [==============================] - 0s 10ms/step - loss: 0.0667 - accuracy:
0.9363
Epoch 3/8
49/49 [==============================] - 0s 10ms/step - loss: 0.0615 - accuracy:
0.9426
Epoch 4/8
49/49 [==============================] - 0s 9ms/step - loss: 0.0582 - accuracy: 0.
9464
Epoch 5/8
49/49 [==============================] - 0s 8ms/step - loss: 0.0546 - accuracy: 0.
9500
Epoch 6/8
49/49 [==============================] - 0s 8ms/step - loss: 0.0528 - accuracy: 0.
9532
Epoch 7/8
49/49 [==============================] - 0s 8ms/step - loss: 0.0507 - accuracy: 0.
9549
Epoch 8/8
49/49 [==============================] - 0s 8ms/step - loss: 0.0498 - accuracy: 0.
9563
782/782 [==============================] - 1s 2ms/step - loss: 0.1112 - accuracy:
0.8802
```

Out[100]:

```
[0.11118891835212708, 0.8801599740982056]
```

In [101…

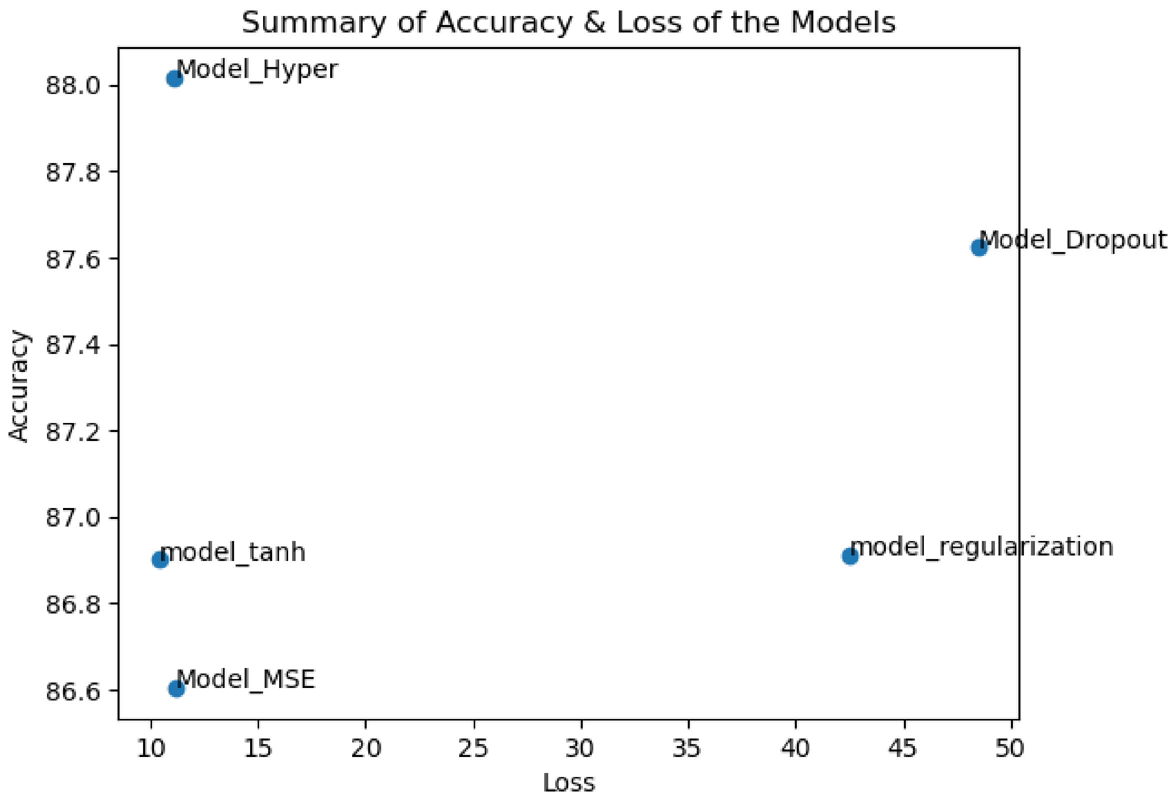
```
#Summary
Models_Loss= np.array([results_Dropout[0],results_Hyper[0],results_MSE[0],results_r
Models_Loss
Models_Accuracy= np.array([results_Dropout[1],results_Hyper[1],results_MSE[1],resul
Models_Accuracy
Labels=['Model_Dropout','Model_Hyper','Model_MSE','model_regularization','model_tar
plt.clf()
```

```
<Figure size 640x480 with 0 Axes>
```

In [102…

```
#compilation
fig, ax = plt.subplots()
ax.scatter(Models_Loss,Models_Accuracy)
for i, txt in enumerate(Labels):
    ax.annotate(txt, (Models_Loss[i],Models_Accuracy[i] ))
plt.title("Summary of Accuracy & Loss of the Models")
plt.ylabel("Accuracy")
plt.xlabel("Loss")

plt.show()
```



In [103…

```
#Summary
#First, data had to be imported, review analysis settings had to be set, and binary
```

In [104…

```
#Conclusion
#Diverse configurations of neural network models exhibited distinct patterns of los
#In the final stage of the trial, we used dropout regularisation to address any cor
```