In [124…

```
!pip install tensorflow
```

```
Requirement already satisfied: tensorflow in c:\users\lenovo\anaconda3\lib\site-packa
ges (2.15.0)
Requirement already satisfied: tensorflow-intel==2.15.0 in c:\users\lenovo\anaconda3
\lib\site-packages (from tensorflow) (2.15.0)
Requirement already satisfied: h5py>=2.9.0 in c:\users\lenovo\anaconda3\lib\site-pack
ages (from tensorflow-intel==2.15.0->tensorflow) (3.6.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\lenovo\anaconda3\lib\sit
e-packages (from tensorflow-intel==2.15.0->tensorflow) (1.6.3)
Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0 in c:\users\lenovo
\anaconda3\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow) (2.15.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\lenovo\anaconda3\lib\site
-packages (from tensorflow-intel==2.15.0->tensorflow) (2.4.0)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\lenovo\anaconda3\lib\site-p
ackages (from tensorflow-intel==2.15.0->tensorflow) (2.1.0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in c:\users\lenovo\anaconda3\lib
\site-packages (from tensorflow-intel==2.15.0->tensorflow) (1.26.4)
Requirement already satisfied: libclang>=13.0.0 in c:\users\lenovo\anaconda3\lib\site
-packages (from tensorflow-intel==2.15.0->tensorflow) (16.0.6)
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\lenovo\anaconda3\lib\site-packages (from tenso
rflow-intel==2.15.0->tensorflow) (4.25.3)
Requirement already satisfied: flatbuffers>=23.5.26 in c:\users\lenovo\anaconda3\lib
\site-packages (from tensorflow-intel==2.15.0->tensorflow) (23.5.26)
Requirement already satisfied: keras<2.16,>=2.15.0 in c:\users\lenovo\anaconda3\lib\s
ite-packages (from tensorflow-intel==2.15.0->tensorflow) (2.15.0)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in c:\users\lenovo
\anaconda3\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow) (0.5.4)
Requirement already satisfied: tensorboard<2.16,>=2.15 in c:\users\lenovo\anaconda3\l
ib\site-packages (from tensorflow-intel==2.15.0->tensorflow) (2.15.2)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in c:\users\lenov
o\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\lenovo\anaconda3\lib\s
ite-packages (from tensorflow-intel==2.15.0->tensorflow) (1.62.0)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\lenovo\anaconda3
\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow) (4.1.1)
Requirement already satisfied: packaging in c:\users\lenovo\anaconda3\lib\site-packag
es (from tensorflow-intel==2.15.0->tensorflow) (21.3)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in c:\users\lenovo\anaconda3\lib\s
ite-packages (from tensorflow-intel==2.15.0->tensorflow) (1.12.1)
Requirement already satisfied: six>=1.12.0 in c:\users\lenovo\anaconda3\lib\site-pack
ages (from tensorflow-intel==2.15.0->tensorflow) (1.16.0)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\lenovo\anaconda3\lib\s
ite-packages (from tensorflow-intel==2.15.0->tensorflow) (0.2.0)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\lenovo\anaconda3\lib\sit
e-packages (from tensorflow-intel==2.15.0->tensorflow) (3.3.0)
Requirement already satisfied: setuptools in c:\users\lenovo\anaconda3\lib\site-packa
ges (from tensorflow-intel==2.15.0->tensorflow) (61.2.0)
Requirement already satisfied: ml-dtypes~=0.2.0 in c:\users\lenovo\anaconda3\lib\site
-packages (from tensorflow-intel==2.15.0->tensorflow) (0.2.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\lenovo\anaconda3\lib\si
te-packages (from astunparse>=1.6.0->tensorflow-intel==2.15.0->tensorflow) (0.37.1)
Requirement already satisfied: google-auth<3,>=1.6.3 in c:\users\lenovo\anaconda3\lib
\site-packages (from tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow)
(2.28.1)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\lenovo\anaconda3\lib\s
ite-packages (from tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (2.
27.1)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in c:\users\leno
vo\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\lenovo\anaconda3\lib\site-
packages (from tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (2.0.3)
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in c:\users\lenovo\anacon
da3\lib\site-packages (from tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensor
flow) (1.2.0)
Requirement already satisfied: markdown>=2.6.8 in c:\users\lenovo\anaconda3\lib\site-
packages (from tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (3.3.4)
Requirement already satisfied: rsa<5,>=3.1.4 in c:\users\lenovo\anaconda3\lib\site-pa
ckages (from google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0
->tensorflow) (4.7.2)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in c:\users\lenovo\anaconda3\li
b\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow-inte
l==2.15.0->tensorflow) (4.2.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\users\lenovo\anaconda3\lib
\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: requests-oauthlib>=0.7.0 in c:\users\lenovo\anaconda3
\lib\site-packages (from google-auth-oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tenso
rflow-intel==2.15.0->tensorflow) (1.3.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in c:\users\lenovo\anaconda3\lib
\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: charset-normalizer~=2.0.0 in c:\users\lenovo\anaconda3
\lib\site-packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow-int
el==2.15.0->tensorflow) (2.0.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\lenovo\anaconda3\lib
\site-packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow-intel==
2.15.0->tensorflow) (1.26.9)
Requirement already satisfied: idna<4,>=2.5 in c:\users\lenovo\anaconda3\lib\site-pac
kages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->t
ensorflow) (3.3)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\lenovo\anaconda3\lib\si
te-packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow-intel==2.1
5.0->tensorflow) (2021.10.8)
Requirement already satisfied: oauthlib>=3.0.0 in c:\users\lenovo\anaconda3\lib\site-
packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<2,>=0.5->tensorboard<2.
16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (3.2.2)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\lenovo\anaconda3
\lib\site-packages (from packaging->tensorflow-intel==2.15.0->tensorflow) (3.0.4)
```

In [132…

```
# The collection of IMDB
# Working with the "IMDB dataset" will include examining a collection of 50,000 intens


# To load the dataset, run the following code:
```

In [126…

```
from tensorflow.keras.datasets import imdb

(train_data, train_labels), (test_data, test_labels) = imdb.load_data(num_words=10000)
```

In [253…

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

```
[list([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, 20
25, 19, 14, 22, 4, 1920, 4613, 469, 4, 22, 71, 87, 12, 16, 43, 530, 38, 76, 15, 13, 1
247, 4, 22, 17, 515, 17, 12, 16, 626, 18, 2, 5, 62, 386, 12, 8, 316, 8, 106, 5, 4, 22
23, 5244, 16, 480, 66, 3785, 33, 4, 130, 12, 16, 38, 619, 5, 25, 124, 51, 36, 135, 4
8, 25, 1415, 33, 6, 22, 12, 215, 28, 77, 52, 5, 14, 407, 16, 82, 2, 8, 4, 107, 117, 5
952, 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, 2
26, 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, 4
55, 9, 45, 43, 38, 1543, 1905, 398, 4, 1649, 26, 6853, 5, 163, 11, 3215, 2, 4, 1153,
9, 194, 775, 7, 8255, 2, 349, 2637, 148, 605, 2, 8003, 15, 123, 125, 68, 2, 6853, 15,
349, 165, 4362, 98, 5, 4, 228, 9, 43, 2, 1157, 15, 299, 120, 5, 120, 174, 11, 220, 17
5, 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, 1382, 9, 8, 168, 145, 23, 4, 1690,
15, 16, 4, 1355, 5, 28, 6, 52, 154, 462, 33, 89, 78, 285, 16, 145, 95])
 list([1, 14, 47, 8, 30, 31, 7, 4, 249, 108, 7, 4, 5974, 54, 61, 369, 13, 71, 149, 1
4, 22, 112, 4, 2401, 311, 12, 16, 3711, 33, 75, 43, 1829, 296, 4, 86, 320, 35, 534, 1
9, 263, 4821, 1301, 4, 1873, 33, 89, 78, 12, 66, 16, 4, 360, 7, 4, 58, 316, 334, 11,
4, 1716, 43, 645, 662, 8, 257, 85, 1200, 42, 1228, 2578, 83, 68, 3912, 15, 36, 165, 1
539, 278, 36, 69, 2, 780, 8, 106, 14, 6905, 1338, 18, 6, 22, 12, 215, 28, 610, 40, 6,
87, 326, 23, 2300, 21, 23, 22, 12, 272, 40, 57, 31, 11, 4, 22, 47, 6, 2307, 51, 9, 17
0, 23, 595, 116, 595, 1352, 13, 191, 79, 638, 89, 2, 14, 9, 8, 106, 607, 624, 35, 53
4, 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, 703, 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, 478
0, 11, 4, 2678, 26, 178, 4, 529, 443, 2, 5, 27, 710, 117, 2, 8123, 165, 47, 84, 37, 1
31, 818, 14, 595, 10, 10, 61, 1242, 1209, 10, 10, 288, 2260, 1702, 34, 2901, 2, 4, 6
5, 496, 4, 231, 7, 790, 5, 6, 320, 234, 2766, 234, 1119, 1574, 7, 496, 4, 139, 929, 2
901, 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, 5
4, 349, 11, 4120, 2959, 45, 58, 1466, 13, 197, 12, 16, 43, 23, 2, 5, 62, 30, 145, 40
2, 11, 4131, 51, 575, 32, 61, 369, 71, 66, 770, 12, 1054, 75, 100, 2198, 8, 4, 105, 3
7, 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, 14, 123, 5, 94
2, 25, 8, 721, 12, 145, 5, 202, 12, 160, 580, 202, 12, 6, 52, 58, 2, 92, 401, 728, 1
2, 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, 168, 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, 221, 109, 29, 1
27, 27, 118, 8, 97, 12, 157, 21, 6789, 2, 9, 6, 66, 78, 1099, 4, 631, 1191, 5, 2642,
272, 191, 1070, 6, 7585, 8, 2197, 2, 2, 544, 5, 383, 1271, 848, 1468, 2, 497, 2, 8, 1
597, 8778, 2, 21, 60, 27, 239, 9, 43, 8368, 209, 405, 10, 10, 12, 764, 40, 4, 248, 2
0, 12, 16, 5, 174, 1791, 72, 7, 51, 6, 1739, 22, 4, 204, 131, 9])] (25000,)
```

In [254…

```
train_data[0]
```

Out[254]:
```
[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]
```

In [128…
```
train_labels[0]
```

Out[128]:
```
1
```

In [255…
```
len(train_labels)
```

Out[255]:
```
25000
```

In [256…
```
test_labels[0]
```

Out[256]:
```
0
```

In [129…
```
max([max(sequence) for sequence in train_data])
```

Out[129]:
```
9999
```

In [133…
```
#Decoding reviews go on to the next
```

In [257…
```
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 [258…
```
decoded_review
```

Out[258]:
```
"? this film was just brilliant casting location scenery story direction everyone's r
eally suited the part they played and you could just imagine being there robert ? is
an amazing actor and now the same being director ? father came from the same scottish
island as myself so i loved the fact there was a real connection with this film the w
itty remarks throughout the film were great it was just brilliant so much that i boug
ht the film as soon as it was released for ? and would recommend it to everyone to wa
tch and the fly fishing was amazing really cried at the end it was so sad and you kno
w what they say if you cry at a film it must have been good and this definitely was a
lso ? to the two little boy's that played the ? of norman and paul they were just bri
lliant children are often left out of the ? list i think because the stars that play
them all grown up are such a big profile for the whole film but these children are am
azing and should be praised for what they have done don't you think the whole story w
as so lovely because it was true and was someone's life after all that was shared wit
h us all"
```

In [135…
```
# Only the top 10,000 most common terms in the training set will be retained, accordin

# The lists of reviews that make up the variables train_data and test_data are lists o

# As a result of our restriction to the top 10,000 most frequently occurring words, no
```

In [259…
```
# Preparation of the Data

# Lists of integers cannot be fed into a neural network. Our lists must be transformed

# 1. One possible solution is to create an integer tensor of the form samples, word_in
# 2. To convert our lists into vectors of 0s and 1s, we could one-hot-encode them. To
# We'll choose the second option. Now let's vectorize our data, which will be done by
```

In [269…
```
import numpy as np

def vectorize_sequences(sequences, dimension=10000):
    # Create an all-zero matrix of shape (len(sequences), dimension)
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1.  # set specific indices of results[i] to 1s
    return results

# Our vectorized training data
x_train = vectorize_sequences(train_data)
# Our vectorized test data
x_test = vectorize_sequences(test_data)
```

In [139…
```
# Currently, this is how our samples appear:
```

In [270…
```
x_train[0]
```

Out[270]:
```
array([0., 1., 1., ..., 0., 0., 0.])
```

In [271…
```
x_test[0]
```

Out[271]:
```
array([0., 1., 1., ..., 0., 0., 0.])
```

In [141…
```
# Vectorizing our labels is something else we ought to do:
```

In [263…
```
# Our labels in vector format
y_train = np.asarray(train_labels).astype('float32')
y_test = np.asarray(test_labels).astype('float32')
```

In [143…
```
# A neural network is now prepared to receive our data.
```

In [144…
```
# Expanding our network

# This is the simplest setup you will ever come across: our labels are scalars (1 and
# Choosing an optimizer and a loss function is the last step. Is it better to use the
```

In [272…
```
from tensorflow import keras
from tensorflow.keras import layers
model = keras.Sequential([
 layers.Dense(16, activation="relu"),
 layers.Dense(16, activation="relu"),
 layers.Dense(1, activation="sigmoid")
])
```

In [273…
```
model.compile(optimizer="rmsprop",
 loss="binary_crossentropy",
 metrics=["accuracy"])
```

In [267…
```
# Verifying our methodology
# We will separate 10,000 samples from the original training data to establish a "vali
```

In [274…
```
x_val = x_train[:10000]
partial_x_train = x_train[10000:]

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

In [148…
```
# We will now train our model in mini-batches of 512 samples for 20 epochs (20 iterati
```

In [275…
```
tf.random.set_seed(1234)

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 [==============================] - 4s 90ms/step - loss: 0.5487 - accuracy: 0.76
48 - val_loss: 0.4410 - val_accuracy: 0.8332
Epoch 2/20
30/30 [==============================] - 1s 22ms/step - loss: 0.3516 - accuracy: 0.88
95 - val_loss: 0.3370 - val_accuracy: 0.8775
Epoch 3/20
30/30 [==============================] - 1s 21ms/step - loss: 0.2611 - accuracy: 0.91
48 - val_loss: 0.2933 - val_accuracy: 0.8867
Epoch 4/20
30/30 [==============================] - 1s 19ms/step - loss: 0.2080 - accuracy: 0.93
42 - val_loss: 0.2801 - val_accuracy: 0.8881
Epoch 5/20
30/30 [==============================] - 1s 20ms/step - loss: 0.1753 - accuracy: 0.94
44 - val_loss: 0.2778 - val_accuracy: 0.8876
Epoch 6/20
30/30 [==============================] - 1s 20ms/step - loss: 0.1463 - accuracy: 0.95
55 - val_loss: 0.2813 - val_accuracy: 0.8870
Epoch 7/20
30/30 [==============================] - 1s 20ms/step - loss: 0.1269 - accuracy: 0.96
13 - val_loss: 0.2894 - val_accuracy: 0.8848
Epoch 8/20
30/30 [==============================] - 1s 22ms/step - loss: 0.1077 - accuracy: 0.96
84 - val_loss: 0.3031 - val_accuracy: 0.8824
Epoch 9/20
30/30 [==============================] - 1s 29ms/step - loss: 0.0922 - accuracy: 0.97
56 - val_loss: 0.3255 - val_accuracy: 0.8825
Epoch 10/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0791 - accuracy: 0.97
95 - val_loss: 0.3369 - val_accuracy: 0.8793
Epoch 11/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0695 - accuracy: 0.98
17 - val_loss: 0.3505 - val_accuracy: 0.8790
Epoch 12/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0549 - accuracy: 0.98
73 - val_loss: 0.4045 - val_accuracy: 0.8690
Epoch 13/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0472 - accuracy: 0.98
95 - val_loss: 0.4064 - val_accuracy: 0.8751
Epoch 14/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0407 - accuracy: 0.99
16 - val_loss: 0.4172 - val_accuracy: 0.8748
Epoch 15/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0335 - accuracy: 0.99
38 - val_loss: 0.4323 - val_accuracy: 0.8748
Epoch 16/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0264 - accuracy: 0.99
55 - val_loss: 0.4512 - val_accuracy: 0.8746
Epoch 17/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0228 - accuracy: 0.99
71 - val_loss: 0.5049 - val_accuracy: 0.8664
Epoch 18/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0183 - accuracy: 0.99
82 - val_loss: 0.4971 - val_accuracy: 0.8745
Epoch 19/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0170 - accuracy: 0.99
71 - val_loss: 0.5205 - val_accuracy: 0.8713
Epoch 20/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0149 - accuracy: 0.99
74 - val_loss: 0.5425 - val_accuracy: 0.8705
```

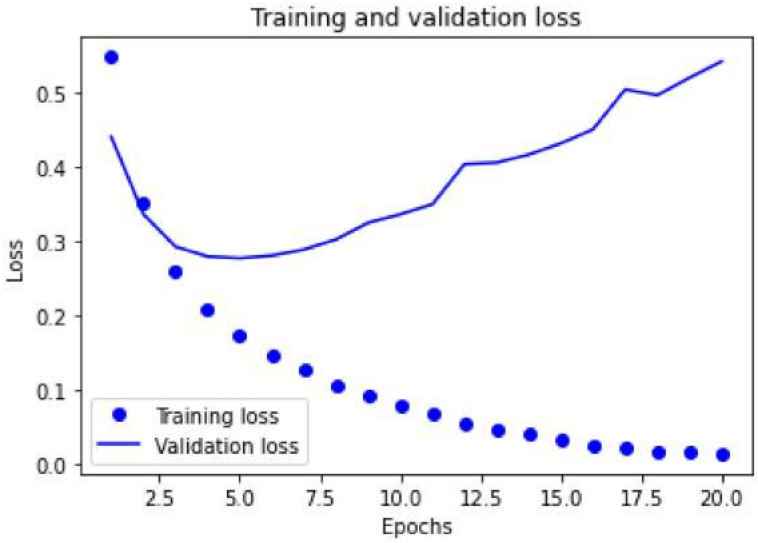
In [150…
```
# Take note that a History object is returned by the model.fit() method. This object h
```

In [276…
```
history_dict = history.history
history_dict.keys()
```

Out[276]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

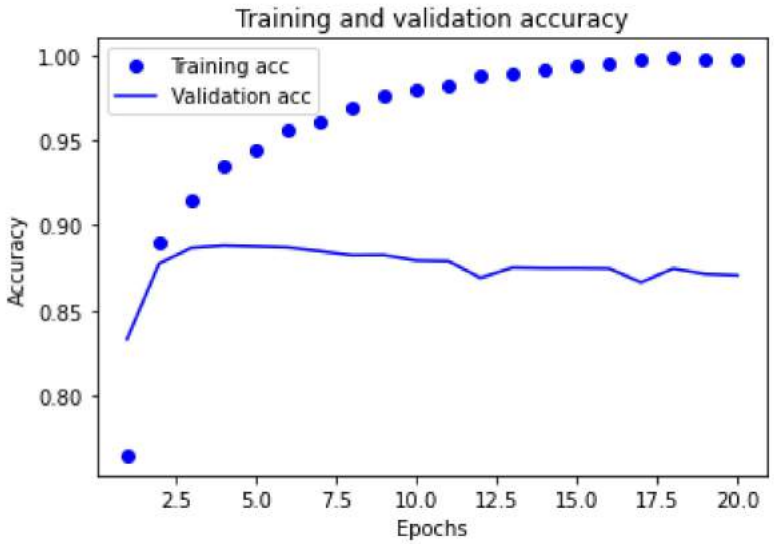
In [277…
```
# There are four entries in total, one for each statistic that was tracked during vali
```

In [279…
```
import matplotlib.pyplot as plt

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

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

# "bo" is for "blue dot"
plt.plot(epochs, loss, 'bo', label='Training loss')
# b is for "solid blue line"
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```



In [280…
```
plt.clf()   # clear figure
acc_values = history_dict['accuracy']
val_acc_values = history_dict['val_accuracy']

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
```

```
plt.show()
```



In [281…
```
#The dots represent training loss and accuracy, whereas the solid lines represent vali
```

In [282…
```
#Model retraining from the scratch
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 [==============================] - 2s 12ms/step - loss: 0.4853 - accuracy: 0.80
18
Epoch 2/4
49/49 [==============================] - 1s 12ms/step - loss: 0.2828 - accuracy: 0.90
30
Epoch 3/4
49/49 [==============================] - 1s 12ms/step - loss: 0.2205 - accuracy: 0.92
07
Epoch 4/4
49/49 [==============================] - 1s 12ms/step - loss: 0.1857 - accuracy: 0.93
31
782/782 [==============================] - 3s 3ms/step - loss: 0.2837 - accuracy: 0.8
868
```

In [283…
```
results
```

Out[283]:
```
[0.2836669087409973, 0.8868399858474731]
```

In [284…
```
#Our basic technique yields 88% accuracy with a loss of 0.2836
```

In [285…
```
#Using a trained model to make predictions about new data
model.predict(x_test)
```

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

Out[285]:
```
array([[0.2051034 ],
       [0.9992665 ],
       [0.87895447],
       ...,
       [0.10239509],
       [0.0861294 ],
       [0.47772923]], dtype=float32)
```

In [286…
```
#Additional Experiments
```

In [ ]:
```
# Two hidden layers were being used. Consider implementing one or three hidden layers
# Try utilizing layers that have fewer or more hidden units: Units: 32, 64...
# Aim to replace binary_crossentropy with the mse loss function.

# Try substituting the tanh activation, which was well-liked in the early days of neur
# Try to improve your model's performance by utilizing any of the techniques you learn
```

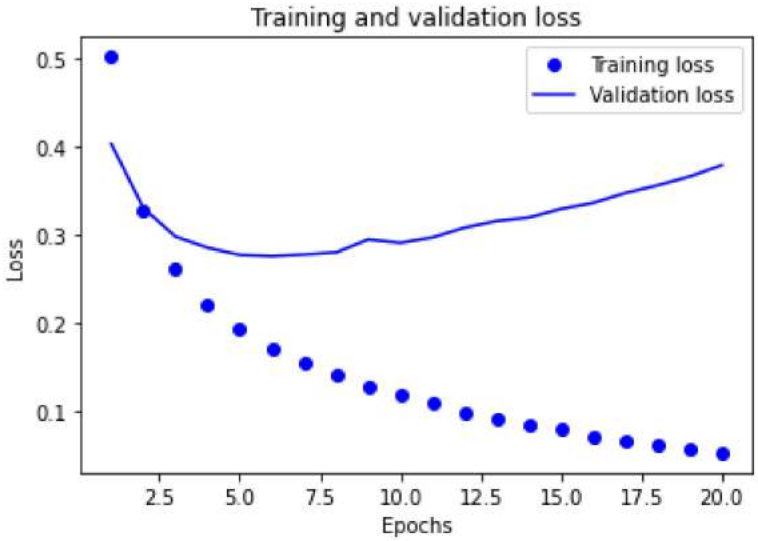
In [287…
```
#Constructing a neural network with single hidden layer
model_1_layer = keras.Sequential([
 layers.Dense(16, activation="relu"),
 layers.Dense(1, activation="sigmoid")
])
model_1_layer.compile(optimizer="rmsprop",
 loss="binary_crossentropy",
 metrics=["accuracy"])
x_val1 = x_train[:10000]
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))
```

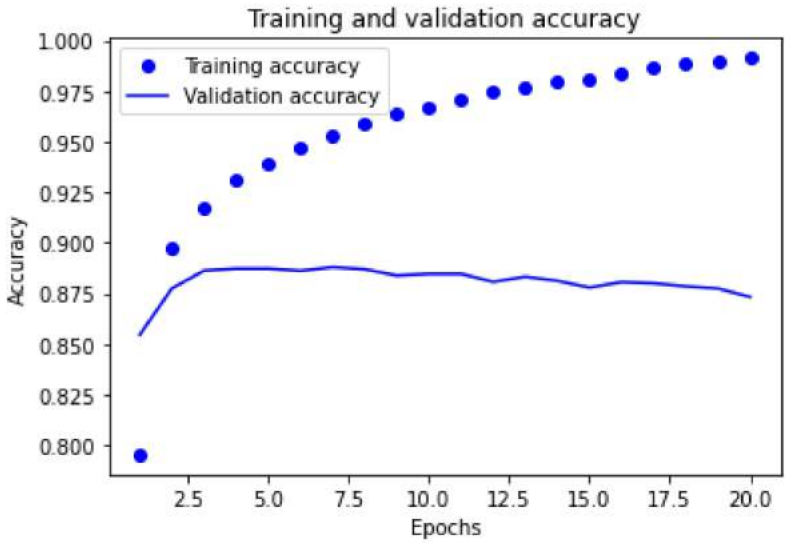
```
Epoch 1/20
30/30 [==============================] - 5s 122ms/step - loss: 0.5027 - accuracy: 0.7
953 - val_loss: 0.4036 - val_accuracy: 0.8546
Epoch 2/20
30/30 [==============================] - 1s 18ms/step - loss: 0.3283 - accuracy: 0.89
71 - val_loss: 0.3304 - val_accuracy: 0.8775
Epoch 3/20
30/30 [==============================] - 1s 22ms/step - loss: 0.2618 - accuracy: 0.91
71 - val_loss: 0.2981 - val_accuracy: 0.8862
Epoch 4/20
30/30 [==============================] - 1s 19ms/step - loss: 0.2200 - accuracy: 0.93
07 - val_loss: 0.2855 - val_accuracy: 0.8872
Epoch 5/20
30/30 [==============================] - 1s 17ms/step - loss: 0.1928 - accuracy: 0.93
89 - val_loss: 0.2772 - val_accuracy: 0.8872
Epoch 6/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1709 - accuracy: 0.94
67 - val_loss: 0.2759 - val_accuracy: 0.8861
Epoch 7/20
30/30 [==============================] - 1s 20ms/step - loss: 0.1541 - accuracy: 0.95
31 - val_loss: 0.2777 - val_accuracy: 0.8879
Epoch 8/20
30/30 [==============================] - 1s 28ms/step - loss: 0.1399 - accuracy: 0.95
88 - val_loss: 0.2801 - val_accuracy: 0.8869
Epoch 9/20
30/30 [==============================] - 1s 22ms/step - loss: 0.1270 - accuracy: 0.96
42 - val_loss: 0.2948 - val_accuracy: 0.8838
Epoch 10/20
30/30 [==============================] - 1s 23ms/step - loss: 0.1178 - accuracy: 0.96
65 - val_loss: 0.2910 - val_accuracy: 0.8846
Epoch 11/20
30/30 [==============================] - 1s 22ms/step - loss: 0.1078 - accuracy: 0.97
03 - val_loss: 0.2971 - val_accuracy: 0.8846
Epoch 12/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0981 - accuracy: 0.97
50 - val_loss: 0.3083 - val_accuracy: 0.8807
Epoch 13/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0906 - accuracy: 0.97
69 - val_loss: 0.3158 - val_accuracy: 0.8832
Epoch 14/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0838 - accuracy: 0.97
99 - val_loss: 0.3198 - val_accuracy: 0.8812
Epoch 15/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0783 - accuracy: 0.98
10 - val_loss: 0.3296 - val_accuracy: 0.8779
Epoch 16/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0711 - accuracy: 0.98
41 - val_loss: 0.3364 - val_accuracy: 0.8806
Epoch 17/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0653 - accuracy: 0.98
67 - val_loss: 0.3476 - val_accuracy: 0.8800
Epoch 18/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0608 - accuracy: 0.98
82 - val_loss: 0.3564 - val_accuracy: 0.8785
Epoch 19/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0565 - accuracy: 0.99
00 - val_loss: 0.3664 - val_accuracy: 0.8774
Epoch 20/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0521 - accuracy: 0.99
14 - val_loss: 0.3792 - val_accuracy: 0.8732
```

In [288…
```
history_dict1 = history1_layer.history
history_dict1.keys()
```

Out[288]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [291…
```
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 and Validation loss
plt.plot(epochs1, loss_value1, "bo", label="Training loss")
plt.plot(epochs1, val_loss_value1, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
#Plotting graph of Training and Validation Accuracy
plt.clf()
accuracy1 = history_dict1["accuracy"]
val_accuracy1 = history_dict1["val_accuracy"]
plt.plot(epochs1, accuracy1, "bo", label="Training accuracy")
plt.plot(epochs1, val_accuracy1, "b", label="Validation accuracy")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```





In [292…
```
#The fewer layers start to overfit later than the reference model, as you can see. Now
```

In [295…
```
#Creating the 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 [==============================] - 2s 16ms/step - loss: 0.4661 - accuracy: 0.81
02
Epoch 2/5
49/49 [==============================] - 1s 16ms/step - loss: 0.2877 - accuracy: 0.90
27
Epoch 3/5
49/49 [==============================] - 1s 12ms/step - loss: 0.2325 - accuracy: 0.91
89
Epoch 4/5
49/49 [==============================] - 1s 14ms/step - loss: 0.2015 - accuracy: 0.92
90
Epoch 5/5
49/49 [==============================] - 1s 12ms/step - loss: 0.1818 - accuracy: 0.93
61
782/782 [==============================] - 3s 4ms/step - loss: 0.2805 - accuracy: 0.8
877
```

In [296…
```
print(result_1_layer)
```

```
[0.28052905201911926, 0.887719988822937]
```

In [297…
```
##The accuracy is 88.78% and the loss on the test set is 0.280%.
```

In [298…
```
model_1_layer.predict(x_test)
```

```
782/782 [==============================] - 3s 4ms/step
```

Out[298]:
```
array([[0.20855999],
       [0.9999166 ],
       [0.80734396],
       ...,
       [0.11771762],
       [0.08556154],
       [0.49819624]], dtype=float32)
```

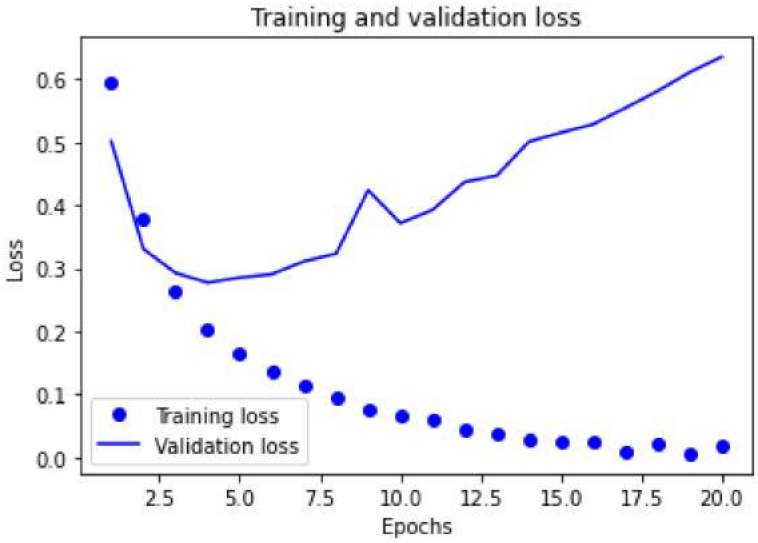
In [299…
```
#Building a neural network with 3 hidden layers
model_3_layers = keras.Sequential([
 layers.Dense(16, activation="relu"),
 layers.Dense(16, activation="relu"),
 layers.Dense(16, activation="relu"),
 layers.Dense(1, activation="sigmoid")
])
model_3_layers.compile(optimizer="rmsprop",
 loss="binary_crossentropy",
 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))
```

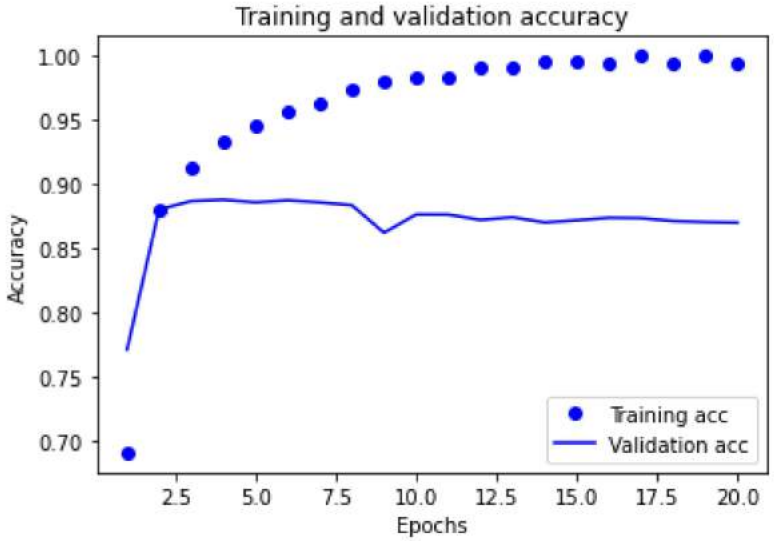
```
Epoch 1/20
30/30 [==============================] - 5s 99ms/step - loss: 0.5943 - accuracy: 0.69
07 - val_loss: 0.5013 - val_accuracy: 0.7710
Epoch 2/20
30/30 [==============================] - 1s 28ms/step - loss: 0.3775 - accuracy: 0.87
92 - val_loss: 0.3304 - val_accuracy: 0.8801
Epoch 3/20
30/30 [==============================] - 1s 19ms/step - loss: 0.2630 - accuracy: 0.91
23 - val_loss: 0.2925 - val_accuracy: 0.8866
Epoch 4/20
30/30 [==============================] - 1s 17ms/step - loss: 0.2028 - accuracy: 0.93
28 - val_loss: 0.2773 - val_accuracy: 0.8876
Epoch 5/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1654 - accuracy: 0.94
59 - val_loss: 0.2852 - val_accuracy: 0.8856
Epoch 6/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1353 - accuracy: 0.95
65 - val_loss: 0.2905 - val_accuracy: 0.8872
Epoch 7/20
30/30 [==============================] - 1s 19ms/step - loss: 0.1151 - accuracy: 0.96
31 - val_loss: 0.3108 - val_accuracy: 0.8856
Epoch 8/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0942 - accuracy: 0.97
29 - val_loss: 0.3231 - val_accuracy: 0.8834
Epoch 9/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0757 - accuracy: 0.97
96 - val_loss: 0.4237 - val_accuracy: 0.8619
Epoch 10/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0660 - accuracy: 0.98
21 - val_loss: 0.3713 - val_accuracy: 0.8762
Epoch 11/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0594 - accuracy: 0.98
33 - val_loss: 0.3925 - val_accuracy: 0.8761
Epoch 12/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0423 - accuracy: 0.99
02 - val_loss: 0.4368 - val_accuracy: 0.8719
Epoch 13/20
30/30 [==============================] - 0s 17ms/step - loss: 0.0366 - accuracy: 0.99
10 - val_loss: 0.4468 - val_accuracy: 0.8740
Epoch 14/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0271 - accuracy: 0.99
53 - val_loss: 0.5007 - val_accuracy: 0.8699
Epoch 15/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0257 - accuracy: 0.99
48 - val_loss: 0.5150 - val_accuracy: 0.8716
Epoch 16/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0247 - accuracy: 0.99
31 - val_loss: 0.5283 - val_accuracy: 0.8736
Epoch 17/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0098 - accuracy: 0.99
93 - val_loss: 0.5540 - val_accuracy: 0.8733
Epoch 18/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0214 - accuracy: 0.99
31 - val_loss: 0.5807 - val_accuracy: 0.8710
Epoch 19/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0059 - accuracy: 0.99
97 - val_loss: 0.6106 - val_accuracy: 0.8702
Epoch 20/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0173 - accuracy: 0.99
43 - val_loss: 0.6352 - val_accuracy: 0.8698
```

In [300…
```
history_dict_3 = history_3_layers.history
history_dict_3.keys()
```

Out[300]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [303…
```
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, "bo", label="Training loss")
plt.plot(epochs3, val_loss_val3, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
plt.clf() #clear figure
accuracy3 = history_dict_3["accuracy"]
val_accuracy3 = history_dict_3["val_accuracy"]
plt.plot(epochs3, accuracy3, "bo", label="Training acc")
plt.plot(epochs3, val_accuracy3, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```





In [304…
```
#As we can see, overfitting occurs with more layers, so let's utilize three epochs.
```

In [305…
```
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 [==============================] - 2s 11ms/step - loss: 0.5366 - accuracy: 0.77
39
Epoch 2/3
49/49 [==============================] - 1s 13ms/step - loss: 0.2946 - accuracy: 0.89
84
Epoch 3/3
49/49 [==============================] - 1s 18ms/step - loss: 0.2150 - accuracy: 0.92
31
782/782 [==============================] - 4s 4ms/step - loss: 0.3094 - accuracy: 0.8
742
```

In [306…
```
print(results_3_layers)
```

```
[0.30936792492866516, 0.874239981174469]
```

In [307…
```
model_3_layers.predict(x_test)
```

```
782/782 [==============================] - 3s 4ms/step
```

Out[307]:
```
array([[0.17146708],
       [0.9998873 ],
       [0.7784519 ],
       ...,
       [0.1218372 ],
       [0.10095341],
       [0.43765363]], dtype=float32)
```

In [308…
```
# The number of layers in the model does not greatly increase its accuracy. But the th
```

In [309…
```
#Building the Neural Network with Three Layers and 32 Hidden Units.
```

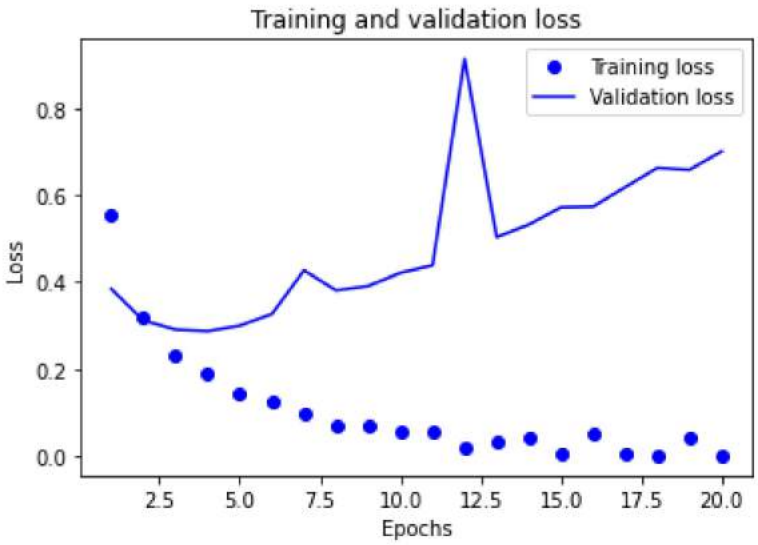
In [311…
```
model_32_units = keras.Sequential([
 layers.Dense(32, activation="relu"),
 layers.Dense(32, activation="relu"),
 layers.Dense(32, activation="relu"),
 layers.Dense(1, activation="sigmoid")
])
#model compilation
model_32_units.compile(optimizer="rmsprop",
 loss="binary_crossentropy",
 metrics=["accuracy"])
#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))
```

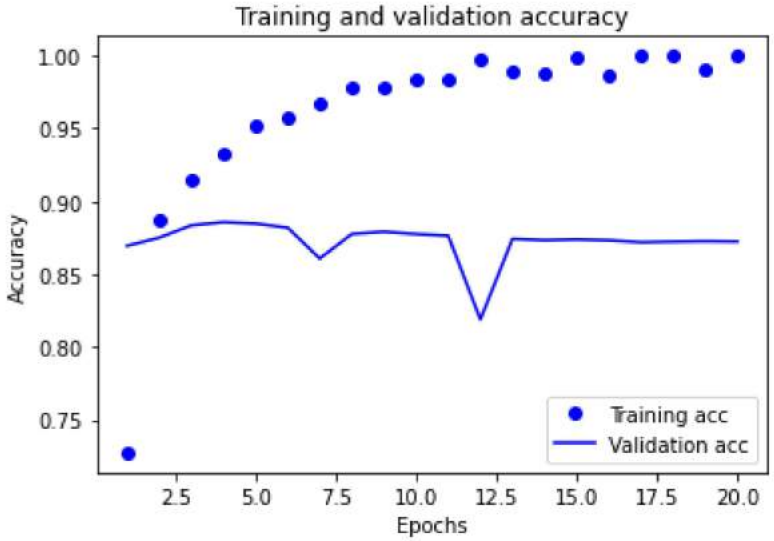
```
Epoch 1/20
30/30 [==============================] - 6s 132ms/step - loss: 0.5540 - accuracy: 0.7
274 - val_loss: 0.3846 - val_accuracy: 0.8694
Epoch 2/20
30/30 [==============================] - 1s 22ms/step - loss: 0.3183 - accuracy: 0.88
73 - val_loss: 0.3119 - val_accuracy: 0.8750
Epoch 3/20
30/30 [==============================] - 1s 29ms/step - loss: 0.2313 - accuracy: 0.91
42 - val_loss: 0.2912 - val_accuracy: 0.8835
Epoch 4/20
30/30 [==============================] - 1s 24ms/step - loss: 0.1880 - accuracy: 0.93
22 - val_loss: 0.2876 - val_accuracy: 0.8857
Epoch 5/20
30/30 [==============================] - 1s 23ms/step - loss: 0.1429 - accuracy: 0.95
17 - val_loss: 0.3001 - val_accuracy: 0.8847
Epoch 6/20
30/30 [==============================] - 1s 31ms/step - loss: 0.1234 - accuracy: 0.95
73 - val_loss: 0.3268 - val_accuracy: 0.8819
Epoch 7/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0997 - accuracy: 0.96
69 - val_loss: 0.4271 - val_accuracy: 0.8607
Epoch 8/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0712 - accuracy: 0.97
89 - val_loss: 0.3810 - val_accuracy: 0.8776
Epoch 9/20
30/30 [==============================] - 1s 29ms/step - loss: 0.0682 - accuracy: 0.97
81 - val_loss: 0.3912 - val_accuracy: 0.8791
Epoch 10/20
30/30 [==============================] - 1s 34ms/step - loss: 0.0565 - accuracy: 0.98
34 - val_loss: 0.4213 - val_accuracy: 0.8775
Epoch 11/20
30/30 [==============================] - 1s 32ms/step - loss: 0.0564 - accuracy: 0.98
38 - val_loss: 0.4385 - val_accuracy: 0.8764
Epoch 12/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0175 - accuracy: 0.99
73 - val_loss: 0.9127 - val_accuracy: 0.8188
Epoch 13/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0349 - accuracy: 0.98
93 - val_loss: 0.5033 - val_accuracy: 0.8742
Epoch 14/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0427 - accuracy: 0.98
83 - val_loss: 0.5322 - val_accuracy: 0.8733
Epoch 15/20
30/30 [==============================] - 1s 29ms/step - loss: 0.0066 - accuracy: 0.99
95 - val_loss: 0.5718 - val_accuracy: 0.8738
Epoch 16/20
30/30 [==============================] - 1s 28ms/step - loss: 0.0505 - accuracy: 0.98
67 - val_loss: 0.5734 - val_accuracy: 0.8732
Epoch 17/20
30/30 [==============================] - 1s 29ms/step - loss: 0.0038 - accuracy: 0.99
98 - val_loss: 0.6180 - val_accuracy: 0.8719
Epoch 18/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0024 - accuracy: 0.99
99 - val_loss: 0.6624 - val_accuracy: 0.8723
Epoch 19/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0410 - accuracy: 0.99
02 - val_loss: 0.6580 - val_accuracy: 0.8726
Epoch 20/20
30/30 [==============================] - 1s 27ms/step - loss: 0.0016 - accuracy: 0.99
99 - val_loss: 0.7002 - val_accuracy: 0.8724
```

In [312…
```
history_dict_32 = history_32_units.history
history_dict_32.keys()
```

Out[312]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [368…
```
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() #clear figure
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 [314…
```
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 [==============================] - 1s 20ms/step - loss: 0.2195 - accuracy: 0.94
13
Epoch 2/3
49/49 [==============================] - 1s 17ms/step - loss: 0.1188 - accuracy: 0.96
14
Epoch 3/3
49/49 [==============================] - 1s 19ms/step - loss: 0.0796 - accuracy: 0.97
40
782/782 [==============================] - 3s 4ms/step - loss: 0.4527 - accuracy: 0.8
675
```

Out[314]:
```
[0.4526526629924774, 0.8675199747085571]
```

In [315…
```
model_32_units.predict(x_test)
```

```
782/782 [==============================] - 4s 4ms/step
```

Out[315]:
```
array([[0.00152871],
       [0.99995714],
       [0.11999177],
       ...,
       [0.00360086],
       [0.01337618],
       [0.8909126 ]], dtype=float32)
```

In [317…
```
# validation set accuracy is 86.75%
```

In [319…
```
# Having the model with 64 units & 2 layers
```

In [320…
```
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
```

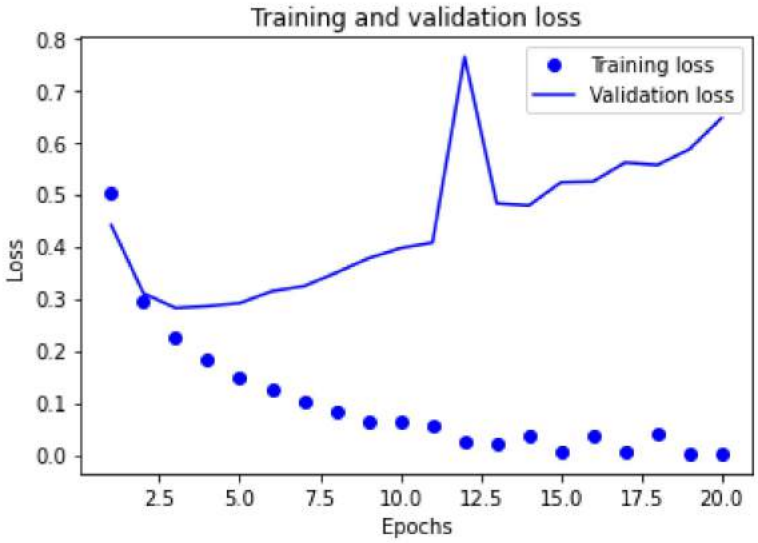
```
x_val_64 = x_train[:10000]
partial_x_train = x_train[10000:]
y_val_64 = y_train[:10000]
partial_y_train = y_train[10000:]
history_64 = model_64_units.fit(partial_x_train,
 partial_y_train,
epochs=20,
batch_size=512,
validation_data=(x_val_64, y_val_64))
```

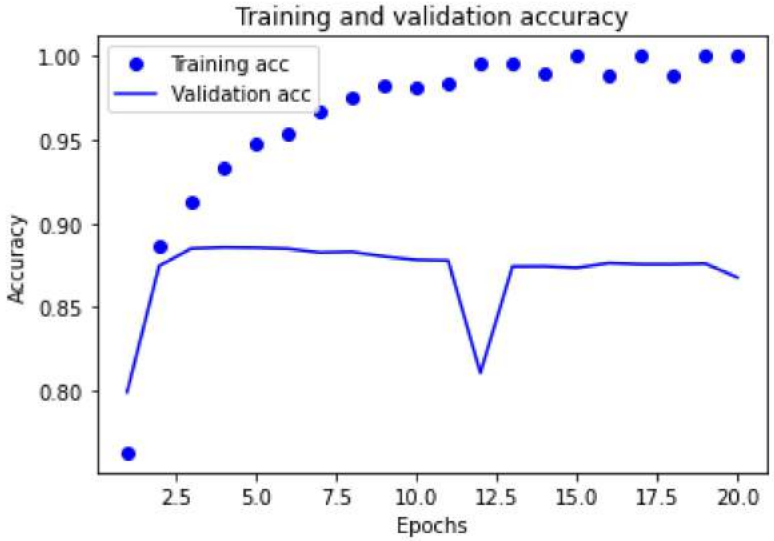
```
Epoch 1/20
30/30 [==============================] - 7s 145ms/step - loss: 0.5029 - accuracy: 0.7
633 - val_loss: 0.4404 - val_accuracy: 0.7994
Epoch 2/20
30/30 [==============================] - 1s 44ms/step - loss: 0.2943 - accuracy: 0.88
65 - val_loss: 0.3099 - val_accuracy: 0.8746
Epoch 3/20
30/30 [==============================] - 2s 54ms/step - loss: 0.2269 - accuracy: 0.91
28 - val_loss: 0.2817 - val_accuracy: 0.8850
Epoch 4/20
30/30 [==============================] - 1s 46ms/step - loss: 0.1821 - accuracy: 0.93
37 - val_loss: 0.2853 - val_accuracy: 0.8857
Epoch 5/20
30/30 [==============================] - 1s 42ms/step - loss: 0.1473 - accuracy: 0.94
80 - val_loss: 0.2910 - val_accuracy: 0.8855
Epoch 6/20
30/30 [==============================] - 1s 44ms/step - loss: 0.1247 - accuracy: 0.95
37 - val_loss: 0.3138 - val_accuracy: 0.8849
Epoch 7/20
30/30 [==============================] - 1s 42ms/step - loss: 0.1030 - accuracy: 0.96
65 - val_loss: 0.3237 - val_accuracy: 0.8826
Epoch 8/20
30/30 [==============================] - 2s 52ms/step - loss: 0.0812 - accuracy: 0.97
55 - val_loss: 0.3491 - val_accuracy: 0.8830
Epoch 9/20
30/30 [==============================] - 1s 42ms/step - loss: 0.0620 - accuracy: 0.98
23 - val_loss: 0.3767 - val_accuracy: 0.8803
Epoch 10/20
30/30 [==============================] - 1s 41ms/step - loss: 0.0632 - accuracy: 0.98
11 - val_loss: 0.3963 - val_accuracy: 0.8782
Epoch 11/20
30/30 [==============================] - 1s 38ms/step - loss: 0.0561 - accuracy: 0.98
32 - val_loss: 0.4071 - val_accuracy: 0.8778
Epoch 12/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0247 - accuracy: 0.99
52 - val_loss: 0.7633 - val_accuracy: 0.8108
Epoch 13/20
30/30 [==============================] - 1s 32ms/step - loss: 0.0218 - accuracy: 0.99
56 - val_loss: 0.4824 - val_accuracy: 0.8743
Epoch 14/20
30/30 [==============================] - 1s 37ms/step - loss: 0.0372 - accuracy: 0.98
91 - val_loss: 0.4783 - val_accuracy: 0.8744
Epoch 15/20
30/30 [==============================] - 2s 53ms/step - loss: 0.0074 - accuracy: 0.99
97 - val_loss: 0.5227 - val_accuracy: 0.8735
Epoch 16/20
30/30 [==============================] - 1s 46ms/step - loss: 0.0376 - accuracy: 0.98
81 - val_loss: 0.5241 - val_accuracy: 0.8763
Epoch 17/20
30/30 [==============================] - 1s 43ms/step - loss: 0.0043 - accuracy: 0.99
99 - val_loss: 0.5607 - val_accuracy: 0.8756
Epoch 18/20
30/30 [==============================] - 1s 46ms/step - loss: 0.0422 - accuracy: 0.98
77 - val_loss: 0.5559 - val_accuracy: 0.8756
Epoch 19/20
30/30 [==============================] - 1s 50ms/step - loss: 0.0030 - accuracy: 0.99
99 - val_loss: 0.5862 - val_accuracy: 0.8760
Epoch 20/20
30/30 [==============================] - 1s 38ms/step - loss: 0.0022 - accuracy: 0.99
99 - val_loss: 0.6460 - val_accuracy: 0.8678
```

In [321…
```
history_dict_64 = history_64.history
history_dict_64.keys()
```

Out[321]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [369…
```
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 [324…
```
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 [==============================] - 2s 31ms/step - loss: 0.2026 - accuracy: 0.94
32
Epoch 2/3
49/49 [==============================] - 2s 31ms/step - loss: 0.1025 - accuracy: 0.96
68
Epoch 3/3
49/49 [==============================] - 1s 28ms/step - loss: 0.0635 - accuracy: 0.98
16
782/782 [==============================] - 4s 5ms/step - loss: 0.4373 - accuracy: 0.8
575
```

Out[324]:
```
[0.4373367130756378, 0.857479989528656]
```

In [328…
```
model_64_units.predict(x_test)
```

```
782/782 [==============================] - 3s 4ms/step
```

Out[328]:
```
array([[0.01078568],
       [0.9999995 ],
       [0.31944314],
       ...,
       [0.01801198],
       [0.00502054],
       [0.53235394]], dtype=float32)
```

In [329…
```
# validation set accuracy = 85.74%
```

In [330…
```
#Training the model with 128 units & 3 layers
```

In [331…
```
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"])
```

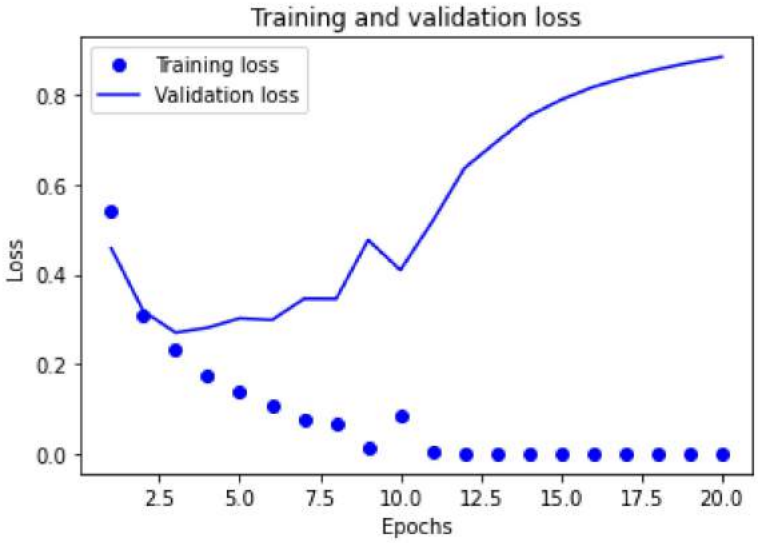
```
# validation
x_val_128 = x_train[:10000]
partial_x_train = x_train[10000:]
y_val_128 = y_train[:10000]
partial_y_train = y_train[10000:]
history_128 = model_128units.fit(partial_x_train,
 partial_y_train,
 epochs=20,
batch_size=512,
validation_data=(x_val_128, y_val_128))
```

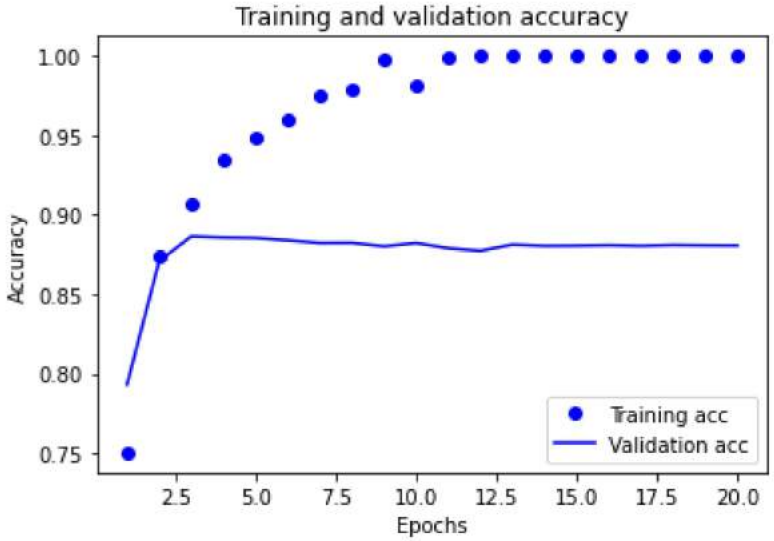
```
Epoch 1/20
30/30 [==============================] - 8s 172ms/step - loss: 0.5436 - accuracy: 0.7
503 - val_loss: 0.4579 - val_accuracy: 0.7933
Epoch 2/20
30/30 [==============================] - 3s 99ms/step - loss: 0.3092 - accuracy: 0.87
41 - val_loss: 0.3179 - val_accuracy: 0.8708
Epoch 3/20
30/30 [==============================] - 2s 81ms/step - loss: 0.2321 - accuracy: 0.90
72 - val_loss: 0.2702 - val_accuracy: 0.8864
Epoch 4/20
30/30 [==============================] - 2s 80ms/step - loss: 0.1722 - accuracy: 0.93
47 - val_loss: 0.2811 - val_accuracy: 0.8857
Epoch 5/20
30/30 [==============================] - 3s 94ms/step - loss: 0.1405 - accuracy: 0.94
85 - val_loss: 0.3021 - val_accuracy: 0.8854
Epoch 6/20
30/30 [==============================] - 3s 93ms/step - loss: 0.1051 - accuracy: 0.96
03 - val_loss: 0.2982 - val_accuracy: 0.8839
Epoch 7/20
30/30 [==============================] - 3s 90ms/step - loss: 0.0765 - accuracy: 0.97
55 - val_loss: 0.3457 - val_accuracy: 0.8822
Epoch 8/20
30/30 [==============================] - 3s 88ms/step - loss: 0.0670 - accuracy: 0.97
86 - val_loss: 0.3454 - val_accuracy: 0.8823
Epoch 9/20
30/30 [==============================] - 3s 84ms/step - loss: 0.0120 - accuracy: 0.99
77 - val_loss: 0.4767 - val_accuracy: 0.8801
Epoch 10/20
30/30 [==============================] - 2s 83ms/step - loss: 0.0840 - accuracy: 0.98
16 - val_loss: 0.4094 - val_accuracy: 0.8822
Epoch 11/20
30/30 [==============================] - 2s 75ms/step - loss: 0.0042 - accuracy: 0.99
95 - val_loss: 0.5183 - val_accuracy: 0.8789
Epoch 12/20
30/30 [==============================] - 2s 72ms/step - loss: 0.0013 - accuracy: 1.00
00 - val_loss: 0.6373 - val_accuracy: 0.8773
Epoch 13/20
30/30 [==============================] - 2s 80ms/step - loss: 5.2434e-04 - accuracy:
1.0000 - val_loss: 0.6961 - val_accuracy: 0.8812
Epoch 14/20
30/30 [==============================] - 2s 78ms/step - loss: 2.5730e-04 - accuracy:
1.0000 - val_loss: 0.7533 - val_accuracy: 0.8804
Epoch 15/20
30/30 [==============================] - 2s 83ms/step - loss: 1.4225e-04 - accuracy:
1.0000 - val_loss: 0.7895 - val_accuracy: 0.8805
Epoch 16/20
30/30 [==============================] - 2s 83ms/step - loss: 9.7533e-05 - accuracy:
1.0000 - val_loss: 0.8182 - val_accuracy: 0.8808
Epoch 17/20
30/30 [==============================] - 3s 85ms/step - loss: 7.4989e-05 - accuracy:
1.0000 - val_loss: 0.8393 - val_accuracy: 0.8804
Epoch 18/20
30/30 [==============================] - 2s 75ms/step - loss: 6.0411e-05 - accuracy:
1.0000 - val_loss: 0.8571 - val_accuracy: 0.8809
Epoch 19/20
30/30 [==============================] - 2s 79ms/step - loss: 5.1089e-05 - accuracy:
1.0000 - val_loss: 0.8725 - val_accuracy: 0.8807
Epoch 20/20
30/30 [==============================] - 2s 71ms/step - loss: 4.3437e-05 - accuracy:
1.0000 - val_loss: 0.8853 - val_accuracy: 0.8806
```

In [332…
```
history_dict_128 = history_128.history
history_dict_128.keys()
```

Out[332]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [370…
```
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 [335…
```
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 [==============================] - 3s 51ms/step - loss: 0.2142 - accuracy: 0.94
16
Epoch 2/2
49/49 [==============================] - 3s 54ms/step - loss: 0.0855 - accuracy: 0.97
23
782/782 [==============================] - 4s 6ms/step - loss: 0.3821 - accuracy: 0.8
707
```

Out[335]:
```
[0.38206371665000916, 0.8707200288772583]
```

In [336…
```
model_128units.predict(x_test)
```

```
782/782 [==============================] - 5s 6ms/step
```

Out[336]:
```
array([[0.02019962],
       [0.9999998 ],
       [0.9758487 ],
       ...,
       [0.0355182 ],
       [0.01845502],
       [0.88924956]], dtype=float32)
```

In [337…
```
#MSE loss function
```

In [338…
```
#with 16 units
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:]
```

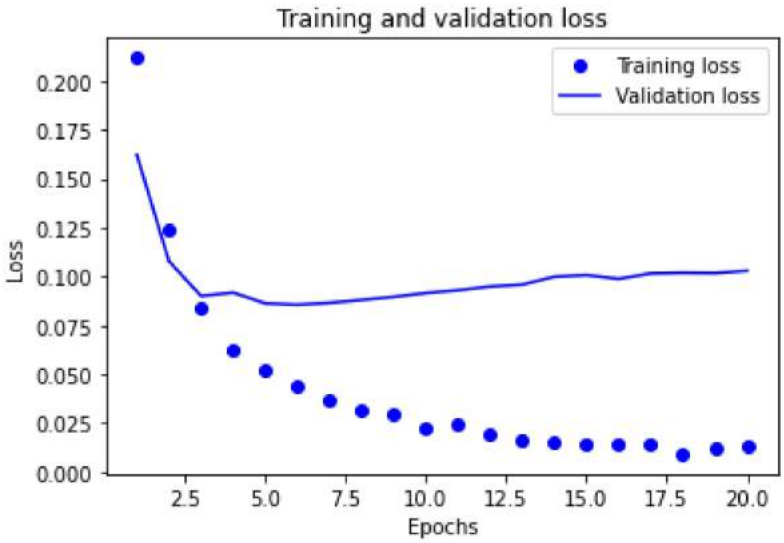
```
y_val_MSE = y_train[:10000]
partial_y_train = y_train[10000:]
# Model Fit
history_MSE = MSE_model.fit(partial_x_train,
 partial_y_train,
epochs=20,
 batch_size=512,
validation_data=(x_val_MSE, y_val_MSE))
```

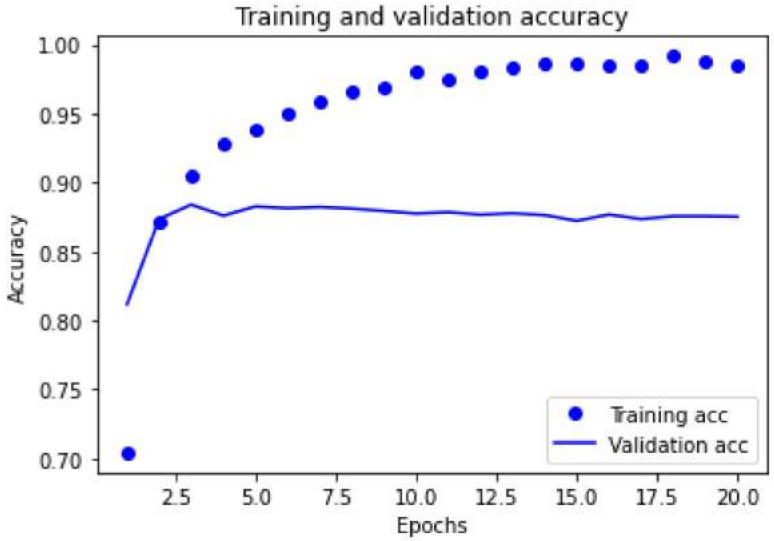
```
Epoch 1/20
30/30 [==============================] - 5s 107ms/step - loss: 0.2118 - accuracy: 0.7
043 - val_loss: 0.1621 - val_accuracy: 0.8119
Epoch 2/20
30/30 [==============================] - 1s 25ms/step - loss: 0.1238 - accuracy: 0.87
19 - val_loss: 0.1078 - val_accuracy: 0.8735
Epoch 3/20
30/30 [==============================] - 1s 40ms/step - loss: 0.0834 - accuracy: 0.90
42 - val_loss: 0.0900 - val_accuracy: 0.8840
Epoch 4/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0627 - accuracy: 0.92
86 - val_loss: 0.0917 - val_accuracy: 0.8760
Epoch 5/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0524 - accuracy: 0.93
89 - val_loss: 0.0863 - val_accuracy: 0.8827
Epoch 6/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0444 - accuracy: 0.94
93 - val_loss: 0.0856 - val_accuracy: 0.8814
Epoch 7/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0371 - accuracy: 0.95
92 - val_loss: 0.0865 - val_accuracy: 0.8823
Epoch 8/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0315 - accuracy: 0.96
67 - val_loss: 0.0880 - val_accuracy: 0.8811
Epoch 9/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0292 - accuracy: 0.96
95 - val_loss: 0.0896 - val_accuracy: 0.8794
Epoch 10/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0226 - accuracy: 0.97
98 - val_loss: 0.0915 - val_accuracy: 0.8775
Epoch 11/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0241 - accuracy: 0.97
43 - val_loss: 0.0930 - val_accuracy: 0.8785
Epoch 12/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0194 - accuracy: 0.98
07 - val_loss: 0.0949 - val_accuracy: 0.8767
Epoch 13/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0167 - accuracy: 0.98
40 - val_loss: 0.0959 - val_accuracy: 0.8777
Epoch 14/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0149 - accuracy: 0.98
66 - val_loss: 0.0999 - val_accuracy: 0.8764
Epoch 15/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0145 - accuracy: 0.98
61 - val_loss: 0.1007 - val_accuracy: 0.8722
Epoch 16/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0143 - accuracy: 0.98
54 - val_loss: 0.0988 - val_accuracy: 0.8768
Epoch 17/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0139 - accuracy: 0.98
51 - val_loss: 0.1015 - val_accuracy: 0.8735
Epoch 18/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0092 - accuracy: 0.99
18 - val_loss: 0.1019 - val_accuracy: 0.8756
Epoch 19/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0119 - accuracy: 0.98
74 - val_loss: 0.1017 - val_accuracy: 0.8756
Epoch 20/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0132 - accuracy: 0.98
56 - val_loss: 0.1029 - val_accuracy: 0.8752
```

In [339…
```
historydict_MSE = history_MSE.history
historydict_MSE.keys()
```

Out[339]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [371…
```
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 [341…
```
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 [==============================] - 1s 12ms/step - loss: 0.0474 - accuracy: 0.94
42
Epoch 2/8
49/49 [==============================] - 1s 12ms/step - loss: 0.0382 - accuracy: 0.95
57
Epoch 3/8
49/49 [==============================] - 1s 14ms/step - loss: 0.0326 - accuracy: 0.96
40
Epoch 4/8
49/49 [==============================] - 1s 12ms/step - loss: 0.0289 - accuracy: 0.96
88
Epoch 5/8
49/49 [==============================] - 1s 11ms/step - loss: 0.0262 - accuracy: 0.97
21
Epoch 6/8
49/49 [==============================] - 1s 13ms/step - loss: 0.0242 - accuracy: 0.97
40
Epoch 7/8
49/49 [==============================] - 1s 13ms/step - loss: 0.0228 - accuracy: 0.97
60
Epoch 8/8
49/49 [==============================] - 1s 18ms/step - loss: 0.0207 - accuracy: 0.97
91
782/782 [==============================] - 3s 4ms/step - loss: 0.1153 - accuracy: 0.8
637
```

Out[341]:
```
[0.11533695459365845, 0.8636800050735474]
```

In [342…
```
MSE_model.predict(x_test)
```

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

Out[342]:
```
array([[0.00543962],
       [0.9999254 ],
       [0.76665753],
       ...,
       [0.27038807],
       [0.00630322],
       [0.9021337 ]], dtype=float32)
```

In [343…
```
#Tanh activation
```

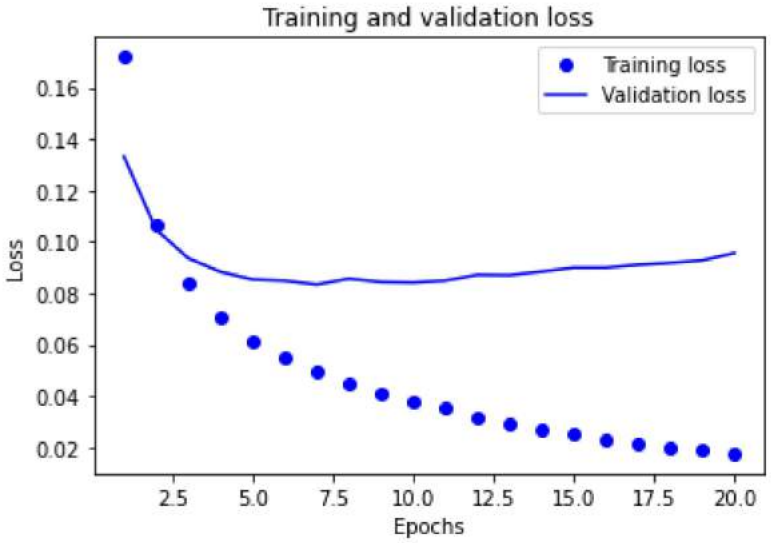
In [344…
```
tanh = keras.Sequential([
 layers.Dense(16, activation="tanh"),
 layers.Dense(1, activation="sigmoid")
])
tanh.compile(optimizer='rmsprop',
 loss='mse',
 metrics=['accuracy'])
x_val_tanh = x_train[:10000]
partial_x_train = x_train[10000:]
y_val_tanh = y_train[:10000]
partial_y_train = y_train[10000:]
historytanh_model = tanh.fit(partial_x_train,
 partial_y_train,
 epochs=20,
batch_size=512,
validation_data=(x_val_tanh, y_val_tanh))
```

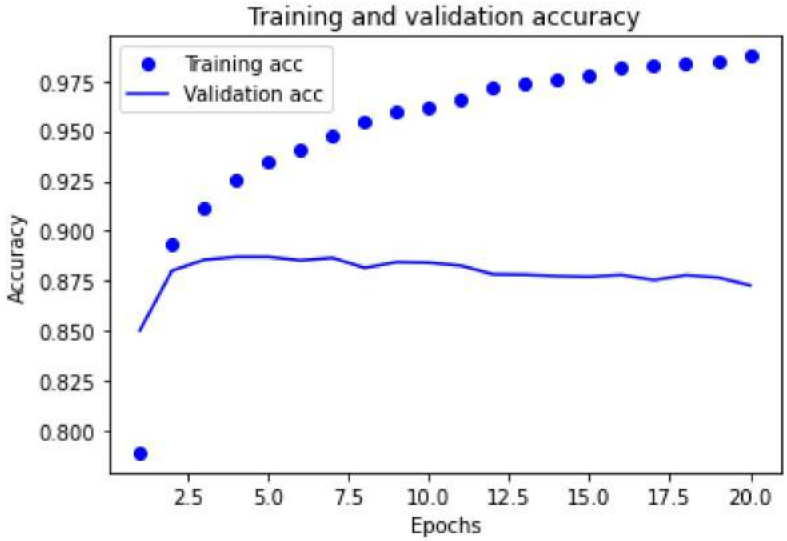
```
Epoch 1/20
30/30 [==============================] - 4s 95ms/step - loss: 0.1719 - accuracy: 0.78
89 - val_loss: 0.1331 - val_accuracy: 0.8501
Epoch 2/20
30/30 [==============================] - 1s 19ms/step - loss: 0.1063 - accuracy: 0.89
32 - val_loss: 0.1047 - val_accuracy: 0.8800
Epoch 3/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0839 - accuracy: 0.91
18 - val_loss: 0.0936 - val_accuracy: 0.8854
Epoch 4/20
30/30 [==============================] - 0s 16ms/step - loss: 0.0704 - accuracy: 0.92
53 - val_loss: 0.0883 - val_accuracy: 0.8869
Epoch 5/20
30/30 [==============================] - 0s 17ms/step - loss: 0.0614 - accuracy: 0.93
48 - val_loss: 0.0854 - val_accuracy: 0.8869
Epoch 6/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0551 - accuracy: 0.94
09 - val_loss: 0.0849 - val_accuracy: 0.8851
Epoch 7/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0497 - accuracy: 0.94
79 - val_loss: 0.0834 - val_accuracy: 0.8864
Epoch 8/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0451 - accuracy: 0.95
44 - val_loss: 0.0857 - val_accuracy: 0.8814
Epoch 9/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0413 - accuracy: 0.96
01 - val_loss: 0.0844 - val_accuracy: 0.8843
Epoch 10/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0379 - accuracy: 0.96
17 - val_loss: 0.0842 - val_accuracy: 0.8840
Epoch 11/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0352 - accuracy: 0.96
59 - val_loss: 0.0849 - val_accuracy: 0.8825
Epoch 12/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0319 - accuracy: 0.97
13 - val_loss: 0.0871 - val_accuracy: 0.8782
Epoch 13/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0291 - accuracy: 0.97
41 - val_loss: 0.0870 - val_accuracy: 0.8780
Epoch 14/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0271 - accuracy: 0.97
63 - val_loss: 0.0884 - val_accuracy: 0.8773
Epoch 15/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0256 - accuracy: 0.97
74 - val_loss: 0.0899 - val_accuracy: 0.8770
Epoch 16/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0230 - accuracy: 0.98
21 - val_loss: 0.0899 - val_accuracy: 0.8779
Epoch 17/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0217 - accuracy: 0.98
33 - val_loss: 0.0911 - val_accuracy: 0.8753
Epoch 18/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0202 - accuracy: 0.98
39 - val_loss: 0.0918 - val_accuracy: 0.8778
Epoch 19/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0192 - accuracy: 0.98
51 - val_loss: 0.0928 - val_accuracy: 0.8766
Epoch 20/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0177 - accuracy: 0.98
76 - val_loss: 0.0956 - val_accuracy: 0.8727
```

In [345…
```
historydict_tanh = historytanh_model.history
historydict_tanh.keys()
```

Out[345]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [372…
```
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 [347…
```
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 [==============================] - 1s 16ms/step - loss: 0.0478 - accuracy: 0.94
23
Epoch 2/8
49/49 [==============================] - 1s 12ms/step - loss: 0.0411 - accuracy: 0.95
16
Epoch 3/8
49/49 [==============================] - 1s 11ms/step - loss: 0.0367 - accuracy: 0.95
96
Epoch 4/8
49/49 [==============================] - 1s 15ms/step - loss: 0.0338 - accuracy: 0.96
42
Epoch 5/8
49/49 [==============================] - 1s 11ms/step - loss: 0.0317 - accuracy: 0.96
72
Epoch 6/8
49/49 [==============================] - 1s 10ms/step - loss: 0.0288 - accuracy: 0.97
06
Epoch 7/8
49/49 [==============================] - 1s 12ms/step - loss: 0.0273 - accuracy: 0.97
26
Epoch 8/8
49/49 [==============================] - 1s 11ms/step - loss: 0.0261 - accuracy: 0.97
44
782/782 [==============================] - 3s 3ms/step - loss: 0.1043 - accuracy: 0.8
666
```

Out[347]:
```
[0.10429728776216507, 0.866599977016449]
```

In [348…
```
#Adam Operator with 16 units and 3-Layers
```

In [349…
```
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',
```

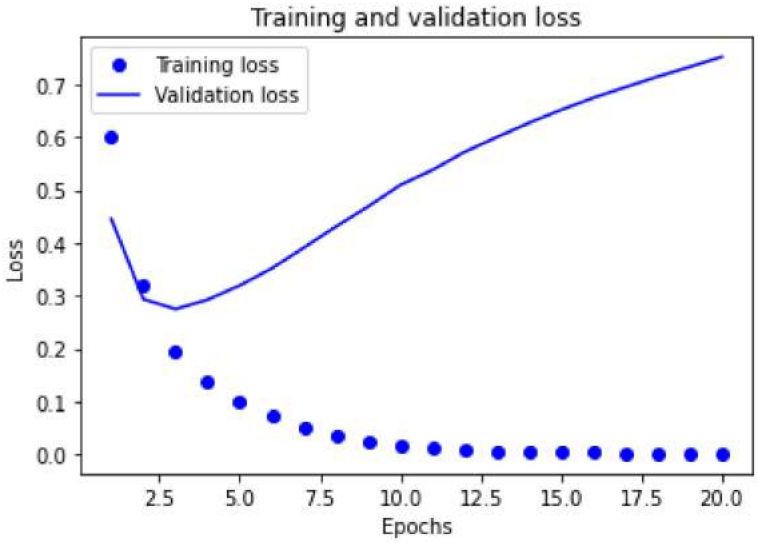
```
 loss='binary_crossentropy',
 metrics=['accuracy'])
x_adam = x_train[:10000]
partial_x_train = x_train[10000:]
y_adam = y_train[:10000]
partial_y_train = y_train[10000:]
historyadam = adam.fit(partial_x_train,
 partial_y_train,
epochs=20,
 batch_size=512,
validation_data=(x_adam, y_adam))
```

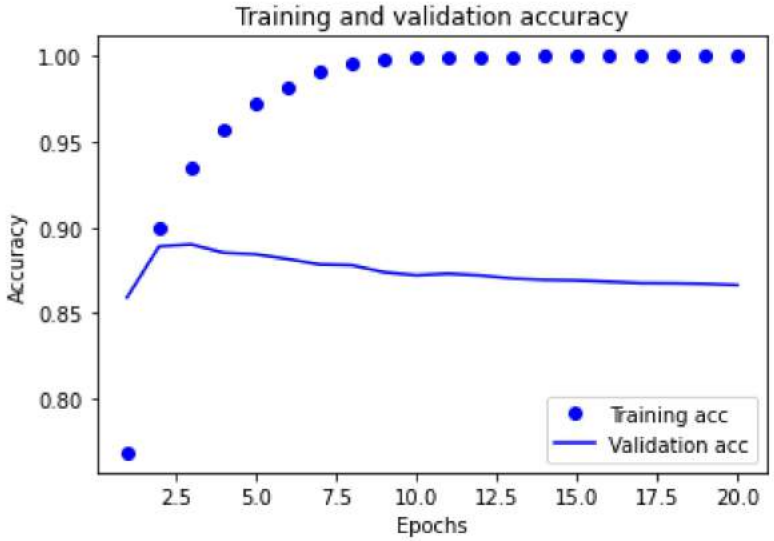
```
Epoch 1/20
30/30 [==============================] - 5s 68ms/step - loss: 0.5995 - accuracy: 0.76
88 - val_loss: 0.4455 - val_accuracy: 0.8593
Epoch 2/20
30/30 [==============================] - 1s 18ms/step - loss: 0.3219 - accuracy: 0.89
95 - val_loss: 0.2937 - val_accuracy: 0.8890
Epoch 3/20
30/30 [==============================] - 1s 17ms/step - loss: 0.1950 - accuracy: 0.93
49 - val_loss: 0.2752 - val_accuracy: 0.8901
Epoch 4/20
30/30 [==============================] - 1s 21ms/step - loss: 0.1363 - accuracy: 0.95
66 - val_loss: 0.2928 - val_accuracy: 0.8853
Epoch 5/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0989 - accuracy: 0.97
25 - val_loss: 0.3198 - val_accuracy: 0.8844
Epoch 6/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0716 - accuracy: 0.98
19 - val_loss: 0.3521 - val_accuracy: 0.8816
Epoch 7/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0490 - accuracy: 0.99
09 - val_loss: 0.3913 - val_accuracy: 0.8784
Epoch 8/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0339 - accuracy: 0.99
52 - val_loss: 0.4312 - val_accuracy: 0.8780
Epoch 9/20
30/30 [==============================] - 1s 27ms/step - loss: 0.0238 - accuracy: 0.99
77 - val_loss: 0.4691 - val_accuracy: 0.8739
Epoch 10/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0169 - accuracy: 0.99
90 - val_loss: 0.5099 - val_accuracy: 0.8722
Epoch 11/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0122 - accuracy: 0.99
94 - val_loss: 0.5382 - val_accuracy: 0.8731
Epoch 12/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0088 - accuracy: 0.99
95 - val_loss: 0.5724 - val_accuracy: 0.8721
Epoch 13/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0066 - accuracy: 0.99
97 - val_loss: 0.6001 - val_accuracy: 0.8703
Epoch 14/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0050 - accuracy: 0.99
99 - val_loss: 0.6272 - val_accuracy: 0.8695
Epoch 15/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0040 - accuracy: 0.99
99 - val_loss: 0.6518 - val_accuracy: 0.8692
Epoch 16/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0032 - accuracy: 0.99
99 - val_loss: 0.6746 - val_accuracy: 0.8684
Epoch 17/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0025 - accuracy: 1.00
00 - val_loss: 0.6948 - val_accuracy: 0.8676
Epoch 18/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0021 - accuracy: 1.00
00 - val_loss: 0.7147 - val_accuracy: 0.8675
Epoch 19/20
30/30 [==============================] - 1s 28ms/step - loss: 0.0018 - accuracy: 1.00
00 - val_loss: 0.7331 - val_accuracy: 0.8672
Epoch 20/20
30/30 [==============================] - 1s 27ms/step - loss: 0.0015 - accuracy: 1.00
00 - val_loss: 0.7522 - val_accuracy: 0.8665
```

In [350…
```
historydict_adam = historyadam.history
historydict_adam.keys()
```

Out[350]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [373…
```
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 [352…
```
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 [==============================] - 1s 13ms/step - loss: 0.2333 - accuracy: 0.93
50
Epoch 2/4
49/49 [==============================] - 1s 17ms/step - loss: 0.1205 - accuracy: 0.96
09
Epoch 3/4
49/49 [==============================] - 1s 15ms/step - loss: 0.0861 - accuracy: 0.97
44
Epoch 4/4
49/49 [==============================] - 1s 14ms/step - loss: 0.0645 - accuracy: 0.98
26
782/782 [==============================] - 3s 4ms/step - loss: 0.5170 - accuracy: 0.8
574
```

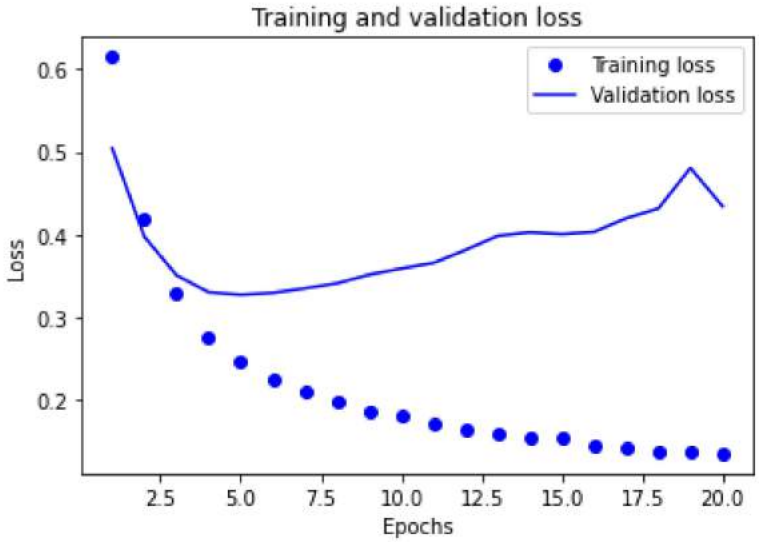
Out[352]:
```
[0.5169916749000549, 0.8574000000953674]
```

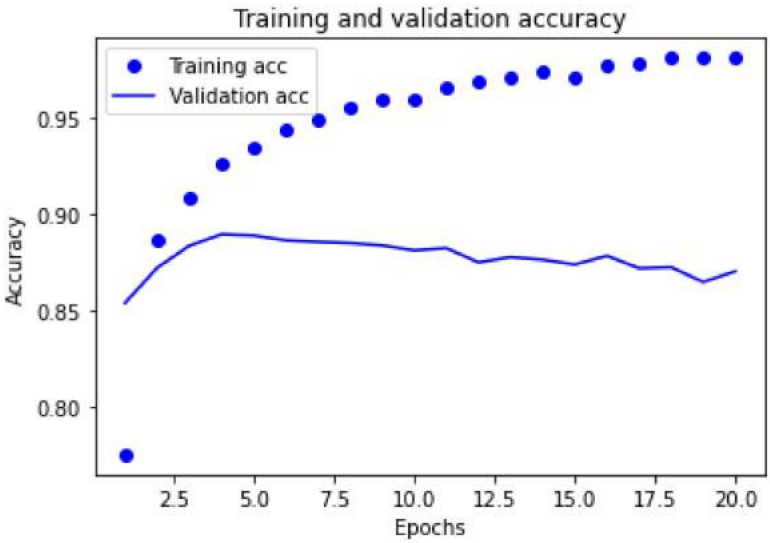
In [353…
```
#Regularization with 16 layers.
from tensorflow.keras import regularizers
regularization = keras.Sequential([
 layers.Dense(16, activation="relu",kernel_regularizer=regularizers.l2(0.001)),
 layers.Dense(16, activation="relu",kernel_regularizer=regularizers.l2(0.001)),
 layers.Dense(1, activation="sigmoid")
])
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 [==============================] - 5s 82ms/step - loss: 0.6149 - accuracy: 0.77
55 - val_loss: 0.5041 - val_accuracy: 0.8540
Epoch 2/20
30/30 [==============================] - 1s 18ms/step - loss: 0.4200 - accuracy: 0.88
61 - val_loss: 0.3971 - val_accuracy: 0.8721
Epoch 3/20
30/30 [==============================] - 1s 21ms/step - loss: 0.3280 - accuracy: 0.90
83 - val_loss: 0.3507 - val_accuracy: 0.8835
Epoch 4/20
30/30 [==============================] - 0s 16ms/step - loss: 0.2760 - accuracy: 0.92
55 - val_loss: 0.3302 - val_accuracy: 0.8894
Epoch 5/20
30/30 [==============================] - 1s 19ms/step - loss: 0.2466 - accuracy: 0.93
47 - val_loss: 0.3272 - val_accuracy: 0.8888
Epoch 6/20
30/30 [==============================] - 1s 21ms/step - loss: 0.2240 - accuracy: 0.94
35 - val_loss: 0.3295 - val_accuracy: 0.8863
Epoch 7/20
30/30 [==============================] - 1s 20ms/step - loss: 0.2095 - accuracy: 0.94
91 - val_loss: 0.3349 - val_accuracy: 0.8855
Epoch 8/20
30/30 [==============================] - 1s 20ms/step - loss: 0.1968 - accuracy: 0.95
53 - val_loss: 0.3410 - val_accuracy: 0.8849
Epoch 9/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1861 - accuracy: 0.95
91 - val_loss: 0.3513 - val_accuracy: 0.8837
Epoch 10/20
30/30 [==============================] - 1s 22ms/step - loss: 0.1813 - accuracy: 0.95
93 - val_loss: 0.3588 - val_accuracy: 0.8811
Epoch 11/20
30/30 [==============================] - 1s 20ms/step - loss: 0.1714 - accuracy: 0.96
49 - val_loss: 0.3655 - val_accuracy: 0.8823
Epoch 12/20
30/30 [==============================] - 1s 17ms/step - loss: 0.1643 - accuracy: 0.96
84 - val_loss: 0.3812 - val_accuracy: 0.8749
Epoch 13/20
30/30 [==============================] - 1s 19ms/step - loss: 0.1598 - accuracy: 0.97
03 - val_loss: 0.3983 - val_accuracy: 0.8776
Epoch 14/20
30/30 [==============================] - 1s 21ms/step - loss: 0.1543 - accuracy: 0.97
36 - val_loss: 0.4026 - val_accuracy: 0.8763
Epoch 15/20
30/30 [==============================] - 1s 21ms/step - loss: 0.1537 - accuracy: 0.97
02 - val_loss: 0.4003 - val_accuracy: 0.8738
Epoch 16/20
30/30 [==============================] - 1s 27ms/step - loss: 0.1453 - accuracy: 0.97
67 - val_loss: 0.4031 - val_accuracy: 0.8783
Epoch 17/20
30/30 [==============================] - 1s 23ms/step - loss: 0.1428 - accuracy: 0.97
74 - val_loss: 0.4196 - val_accuracy: 0.8719
Epoch 18/20
30/30 [==============================] - 1s 25ms/step - loss: 0.1368 - accuracy: 0.98
06 - val_loss: 0.4314 - val_accuracy: 0.8724
Epoch 19/20
30/30 [==============================] - 1s 22ms/step - loss: 0.1362 - accuracy: 0.98
06 - val_loss: 0.4804 - val_accuracy: 0.8648
Epoch 20/20
30/30 [==============================] - 1s 22ms/step - loss: 0.1349 - accuracy: 0.98
07 - val_loss: 0.4341 - val_accuracy: 0.8703
```

Out[353]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [374…
```
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 [355…
```
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 [==============================] - 1s 14ms/step - loss: 0.2531 - accuracy: 0.93
68
Epoch 2/8
49/49 [==============================] - 1s 14ms/step - loss: 0.2084 - accuracy: 0.94
87
Epoch 3/8
49/49 [==============================] - 1s 14ms/step - loss: 0.1970 - accuracy: 0.94
95
Epoch 4/8
49/49 [==============================] - 1s 14ms/step - loss: 0.1839 - accuracy: 0.95
56
Epoch 5/8
49/49 [==============================] - 1s 14ms/step - loss: 0.1802 - accuracy: 0.95
76
Epoch 6/8
49/49 [==============================] - 1s 14ms/step - loss: 0.1735 - accuracy: 0.95
89
Epoch 7/8
49/49 [==============================] - 1s 14ms/step - loss: 0.1688 - accuracy: 0.96
23
Epoch 8/8
49/49 [==============================] - 1s 14ms/step - loss: 0.1683 - accuracy: 0.96
26
782/782 [==============================] - 4s 5ms/step - loss: 0.5084 - accuracy: 0.8
561
```

Out[355]:
```
[0.5084078311920166, 0.8560799956321716]
```

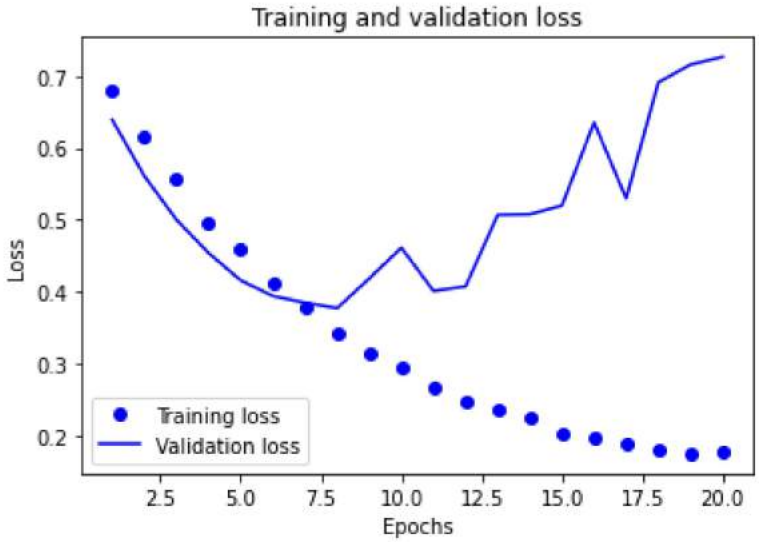
In [357…
```
# Dropout function with 16 units and 3-Layers
```

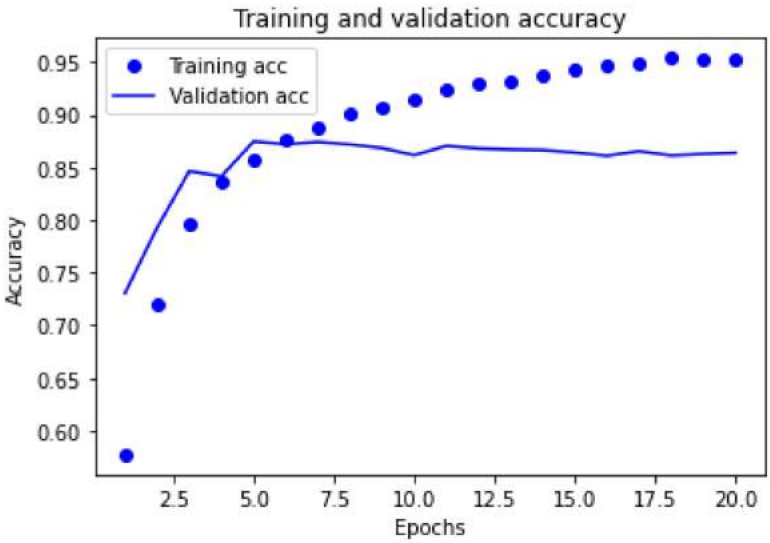
In [358…
```
##Dropout
from tensorflow.keras import regularizers
Dropout = keras.Sequential([
 layers.Dense(16, activation="relu"),
 layers.Dropout(0.5),
 layers.Dense(16, activation="relu"),
 layers.Dropout(0.5),
 layers.Dense(16, activation="relu"),
 layers.Dropout(0.5),layers.Dense(1, activation="sigmoid")
])
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 [==============================] - 5s 64ms/step - loss: 0.6791 - accuracy: 0.57
71 - val_loss: 0.6391 - val_accuracy: 0.7307
Epoch 2/20
30/30 [==============================] - 1s 21ms/step - loss: 0.6153 - accuracy: 0.71
95 - val_loss: 0.5608 - val_accuracy: 0.7928
Epoch 3/20
30/30 [==============================] - 1s 20ms/step - loss: 0.5577 - accuracy: 0.79
63 - val_loss: 0.5001 - val_accuracy: 0.8465
Epoch 4/20
30/30 [==============================] - 0s 17ms/step - loss: 0.4961 - accuracy: 0.83
64 - val_loss: 0.4538 - val_accuracy: 0.8416
Epoch 5/20
30/30 [==============================] - 1s 19ms/step - loss: 0.4582 - accuracy: 0.85
82 - val_loss: 0.4159 - val_accuracy: 0.8747
Epoch 6/20
30/30 [==============================] - 1s 18ms/step - loss: 0.4129 - accuracy: 0.87
59 - val_loss: 0.3941 - val_accuracy: 0.8721
Epoch 7/20
30/30 [==============================] - 1s 19ms/step - loss: 0.3780 - accuracy: 0.88
71 - val_loss: 0.3849 - val_accuracy: 0.8742
Epoch 8/20
30/30 [==============================] - 1s 18ms/step - loss: 0.3430 - accuracy: 0.90
07 - val_loss: 0.3773 - val_accuracy: 0.8719
Epoch 9/20
30/30 [==============================] - 1s 19ms/step - loss: 0.3156 - accuracy: 0.90
62 - val_loss: 0.4185 - val_accuracy: 0.8684
Epoch 10/20
30/30 [==============================] - 1s 18ms/step - loss: 0.2949 - accuracy: 0.91
48 - val_loss: 0.4612 - val_accuracy: 0.8617
Epoch 11/20
30/30 [==============================] - 1s 18ms/step - loss: 0.2660 - accuracy: 0.92
43 - val_loss: 0.4012 - val_accuracy: 0.8705
Epoch 12/20
30/30 [==============================] - 0s 17ms/step - loss: 0.2465 - accuracy: 0.92
90 - val_loss: 0.4075 - val_accuracy: 0.8680
Epoch 13/20
30/30 [==============================] - 1s 24ms/step - loss: 0.2349 - accuracy: 0.93
18 - val_loss: 0.5070 - val_accuracy: 0.8671
Epoch 14/20
30/30 [==============================] - 1s 18ms/step - loss: 0.2264 - accuracy: 0.93
71 - val_loss: 0.5077 - val_accuracy: 0.8665
Epoch 15/20
30/30 [==============================] - 1s 21ms/step - loss: 0.2039 - accuracy: 0.94
39 - val_loss: 0.5200 - val_accuracy: 0.8642
Epoch 16/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1980 - accuracy: 0.94
61 - val_loss: 0.6356 - val_accuracy: 0.8611
Epoch 17/20
30/30 [==============================] - 1s 23ms/step - loss: 0.1887 - accuracy: 0.94
84 - val_loss: 0.5298 - val_accuracy: 0.8655
Epoch 18/20
30/30 [==============================] - 1s 24ms/step - loss: 0.1791 - accuracy: 0.95
41 - val_loss: 0.6912 - val_accuracy: 0.8614
Epoch 19/20
30/30 [==============================] - 1s 39ms/step - loss: 0.1750 - accuracy: 0.95
35 - val_loss: 0.7159 - val_accuracy: 0.8628
Epoch 20/20
30/30 [==============================] - 1s 24ms/step - loss: 0.1765 - accuracy: 0.95
19 - val_loss: 0.7267 - val_accuracy: 0.8639
```

Out[358]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [376…
```
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 [360…
```
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 22ms/step - loss: 0.3767 - accuracy: 0.89
45
Epoch 2/8
49/49 [==============================] - 1s 16ms/step - loss: 0.3116 - accuracy: 0.91
12
Epoch 3/8
49/49 [==============================] - 1s 16ms/step - loss: 0.2722 - accuracy: 0.91
83
Epoch 4/8
49/49 [==============================] - 1s 16ms/step - loss: 0.2603 - accuracy: 0.92
20
Epoch 5/8
49/49 [==============================] - 1s 16ms/step - loss: 0.2467 - accuracy: 0.92
84
Epoch 6/8
49/49 [==============================] - 1s 16ms/step - loss: 0.2333 - accuracy: 0.93
10
Epoch 7/8
49/49 [==============================] - 1s 15ms/step - loss: 0.2170 - accuracy: 0.93
61
Epoch 8/8
49/49 [==============================] - 1s 17ms/step - loss: 0.2067 - accuracy: 0.93
92
782/782 [==============================] - 4s 4ms/step - loss: 0.5213 - accuracy: 0.8
628
```

Out[360]:
```
[0.5213152170181274, 0.8628000020980835]
```

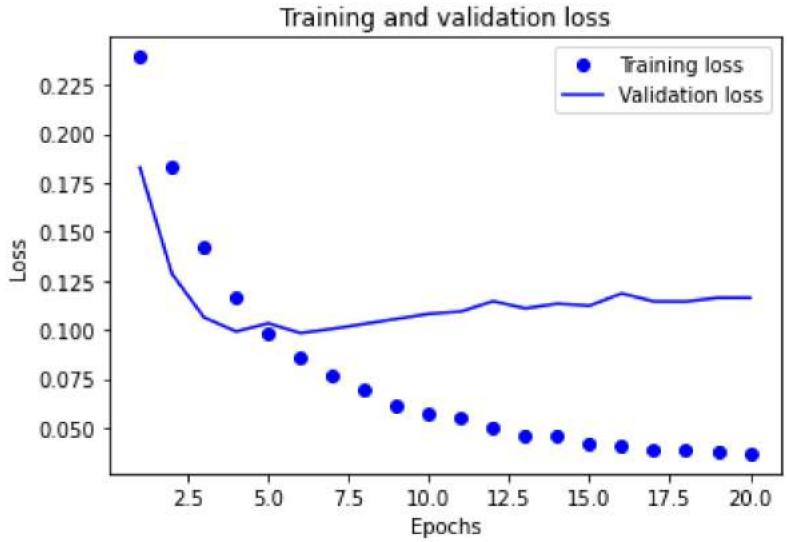
In [361…
```
# Training the model with three layers and 32 units of hypertuned parameters
```

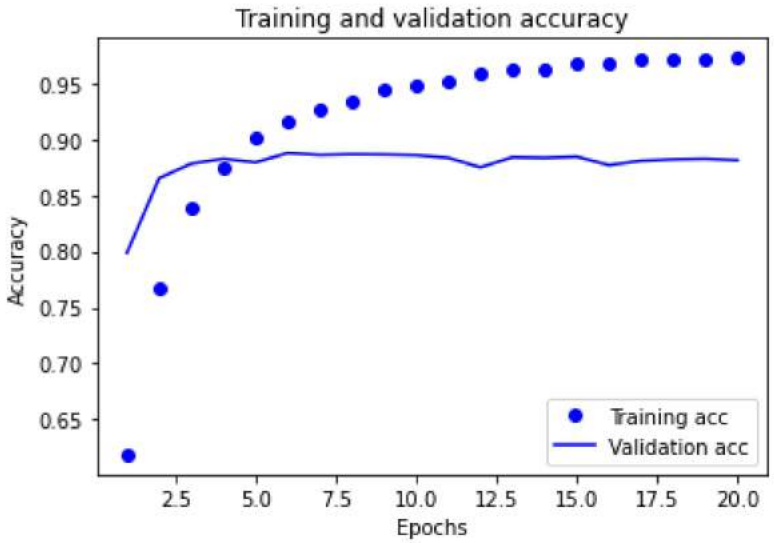
In [362…
```
#Training model with hyper tuned parameters
from tensorflow.keras import regularizers
Hyper = keras.Sequential([
 layers.Dense(32, activation="relu",kernel_regularizer=regularizers.l2(0.0001)),
 layers.Dropout(0.5),
 layers.Dense(32, activation="relu",kernel_regularizer=regularizers.l2(0.0001)),
 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 [==============================] - 5s 89ms/step - loss: 0.2392 - accuracy: 0.61
85 - val_loss: 0.1828 - val_accuracy: 0.7989
Epoch 2/20
30/30 [==============================] - 1s 26ms/step - loss: 0.1826 - accuracy: 0.76
77 - val_loss: 0.1285 - val_accuracy: 0.8653
Epoch 3/20
30/30 [==============================] - 1s 27ms/step - loss: 0.1420 - accuracy: 0.83
93 - val_loss: 0.1062 - val_accuracy: 0.8783
Epoch 4/20
30/30 [==============================] - 1s 47ms/step - loss: 0.1169 - accuracy: 0.87
41 - val_loss: 0.0991 - val_accuracy: 0.8827
Epoch 5/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0977 - accuracy: 0.90
22 - val_loss: 0.1032 - val_accuracy: 0.8794
Epoch 6/20
30/30 [==============================] - 1s 28ms/step - loss: 0.0858 - accuracy: 0.91
62 - val_loss: 0.0983 - val_accuracy: 0.8878
Epoch 7/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0765 - accuracy: 0.92
62 - val_loss: 0.1004 - val_accuracy: 0.8861
Epoch 8/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0698 - accuracy: 0.93
40 - val_loss: 0.1029 - val_accuracy: 0.8867
Epoch 9/20
30/30 [==============================] - 1s 32ms/step - loss: 0.0617 - accuracy: 0.94
43 - val_loss: 0.1055 - val_accuracy: 0.8865
Epoch 10/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0576 - accuracy: 0.94
81 - val_loss: 0.1081 - val_accuracy: 0.8859
Epoch 11/20
30/30 [==============================] - 1s 39ms/step - loss: 0.0550 - accuracy: 0.95
15 - val_loss: 0.1093 - val_accuracy: 0.8835
Epoch 12/20
30/30 [==============================] - 1s 34ms/step - loss: 0.0499 - accuracy: 0.95
81 - val_loss: 0.1145 - val_accuracy: 0.8750
Epoch 13/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0464 - accuracy: 0.96
25 - val_loss: 0.1109 - val_accuracy: 0.8840
Epoch 14/20
30/30 [==============================] - 1s 36ms/step - loss: 0.0458 - accuracy: 0.96
33 - val_loss: 0.1133 - val_accuracy: 0.8835
Epoch 15/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0422 - accuracy: 0.96
75 - val_loss: 0.1122 - val_accuracy: 0.8845
Epoch 16/20
30/30 [==============================] - 1s 32ms/step - loss: 0.0410 - accuracy: 0.96
75 - val_loss: 0.1185 - val_accuracy: 0.8770
Epoch 17/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0389 - accuracy: 0.97
19 - val_loss: 0.1144 - val_accuracy: 0.8808
Epoch 18/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0384 - accuracy: 0.97
13 - val_loss: 0.1144 - val_accuracy: 0.8821
Epoch 19/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0378 - accuracy: 0.97
18 - val_loss: 0.1163 - val_accuracy: 0.8826
Epoch 20/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0367 - accuracy: 0.97
27 - val_loss: 0.1162 - val_accuracy: 0.8814
```

Out[362]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [377…
```
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 [364…
```
Hyper.fit(x_train, y_train, epochs=8, batch_size=512)
results_Hyper = Hyper.evaluate(x_test, y_test)
results_Hyper
```

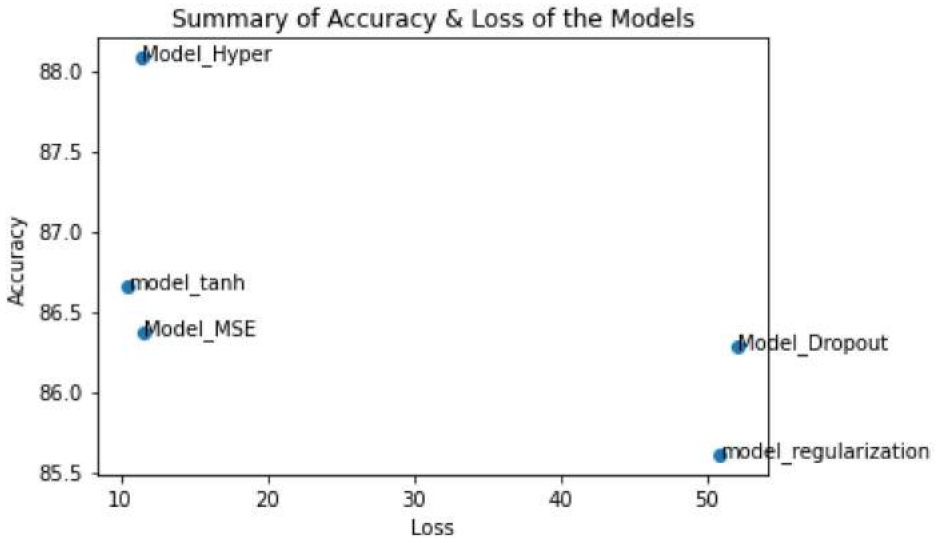
```
Epoch 1/8
49/49 [==============================] - 1s 23ms/step - loss: 0.0727 - accuracy: 0.92
89
Epoch 2/8
49/49 [==============================] - 1s 21ms/step - loss: 0.0662 - accuracy: 0.93
51
Epoch 3/8
49/49 [==============================] - 1s 18ms/step - loss: 0.0600 - accuracy: 0.94
43
Epoch 4/8
49/49 [==============================] - 1s 23ms/step - loss: 0.0568 - accuracy: 0.94
74
Epoch 5/8
49/49 [==============================] - 1s 21ms/step - loss: 0.0535 - accuracy: 0.95
18
Epoch 6/8
49/49 [==============================] - 1s 19ms/step - loss: 0.0521 - accuracy: 0.95
42
Epoch 7/8
49/49 [==============================] - 1s 20ms/step - loss: 0.0496 - accuracy: 0.95
72
Epoch 8/8
49/49 [==============================] - 1s 18ms/step - loss: 0.0475 - accuracy: 0.95
94
782/782 [==============================] - 3s 4ms/step - loss: 0.1145 - accuracy: 0.8
808
```

Out[364]:
```
[0.11451728641986847, 0.8808000087738037]
```

In [365…
```
#Summary

Models_Loss= np.array([results_Dropout[0],results_Hyper[0],results_MSE[0],results_regu
Models_Loss
Models_Accuracy= np.array([results_Dropout[1],results_Hyper[1],results_MSE[1],results_
Models_Accuracy
Labels=['Model_Dropout','Model_Hyper','Model_MSE','model_regularization','model_tanh']
plt.clf()
```

```
<Figure size 432x288 with 0 Axes>
```

In [366…
```
#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 [181…
```
# Summary
# Once the data was imported, we adjusted the word count and review scope parameters t
# Next, we looked at the consequences of altering the amount of hidden units—32 and 64
```

Out[181]:
```
[0.3147193491458893, 0.8758000135421753]
```

In [182…
```
# Conclusion
# Differently configured neural network models displayed varying patterns of loss and
# By including dropout layers into a new model and running training and evaluation on
# The use of dropout regularization significantly improved the models' overall perform
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: