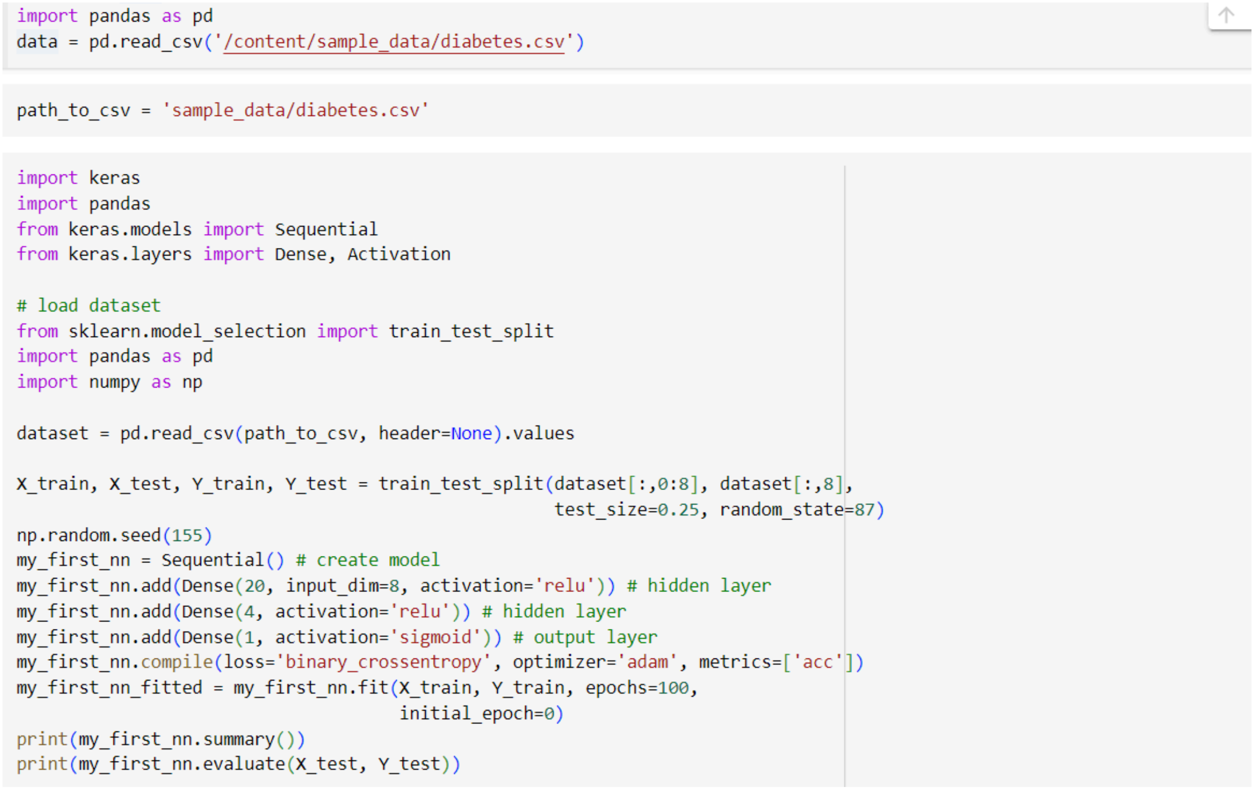
Spring 2024: CS5720 Neural Networks & Deep Learning - ICP-6
Assignment-6
NAME:Vinay Kumar Reddy Gunuguntla
STUDENT ID:700745726

Github Link: https://github.com/VinayGunuguntla/icp6.git
Video Link:
https://drive.google.com/file/d/1DCbZ6Nn1gwebKM1dXT79DFIakYn62jh0/view?usp=drive_link

In class programming: 1. UIn class programming:
1. Use the use case in the class:
a. Add more Dense layers to the existing code and check how the accuracy changes.
2. Change the data source to Breast Cancer dataset * available in the source code folder and make required
changes. Report accuracy of the model.
3. Normalize the data before feeding the data to the model and check how the normalization change your
accuracy (code given below).
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()se the use case in the class: a. Add more Dense layers to the existing code and check how the accuracy changes. 2. Change the data source to Breast Cancer dataset * available in the source code folder and make required changes. Report accuracy of the model. 3. Normalize the data before feeding the data to the model and check how the normalization change your accuracy (code given below). from sklearn.preprocessing import StandardScaler sc = StandardScaler()

#CODE

Code:



```
import pandas as pd
data = pd.read_csv('/content/sample_data/diabetes.csv')
```

```
path_to_csv = 'sample_data/diabetes.csv'
```

```
import keras
import pandas
from keras.models import Sequential
from keras.layers import Dense, Activation

# load dataset
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np

dataset = pd.read_csv(path_to_csv, header=None).values

X_train, X_test, Y_train, Y_test = train_test_split(dataset[:,0:8], dataset[:,8],
                                                    test_size=0.25, random_state=87)
np.random.seed(155)
my_first_nn = Sequential() # create model
my_first_nn.add(Dense(20, input_dim=8, activation='relu')) # hidden layer
my_first_nn.add(Dense(4, activation='relu')) # hidden layer
my_first_nn.add(Dense(1, activation='sigmoid')) # output layer
my_first_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
my_first_nn_fitted = my_first_nn.fit(X_train, Y_train, epochs=100,
                                     initial_epoch=0)
print(my_first_nn.summary())
print(my_first_nn.evaluate(X_test, Y_test))
```

Output:

```
Epoch 1/100
18/18 [==============================] - 1s 3ms/step - loss: 3.5958 - acc:
0.6597
Epoch 2/100
18/18 [==============================] - 0s 2ms/step - loss: 2.5002 - acc:
0.6337
Epoch 3/100
18/18 [==============================] - 0s 2ms/step - loss: 1.9136 - acc:
0.5312
Epoch 4/100
18/18 [==============================] - 0s 3ms/step - loss: 1.5350 - acc:
0.5087
Epoch 5/100
18/18 [==============================] - 0s 2ms/step - loss: 1.2199 - acc:
0.4722
Epoch 6/100
18/18 [==============================] - 0s 2ms/step - loss: 0.9732 - acc:
0.4149
Epoch 7/100
18/18 [==============================] - 0s 2ms/step - loss: 0.8484 - acc:
0.3455
Epoch 8/100
```

```
18/18 [==============================] - 0s 2ms/step - loss: 0.7990 - acc:
0.4913
Epoch 9/100
18/18 [==============================] - 0s 2ms/step - loss: 0.7753 - acc:
0.6667
Epoch 10/100
18/18 [==============================] - 0s 2ms/step - loss: 0.7636 - acc:
0.6597
Epoch 11/100
18/18 [==============================] - 0s 2ms/step - loss: 0.7619 - acc:
0.6632
Epoch 12/100
18/18 [==============================] - 0s 2ms/step - loss: 0.7461 - acc:
0.6562
Epoch 13/100
18/18 [==============================] - 0s 2ms/step - loss: 0.7265 - acc:
0.6615
Epoch 14/100
18/18 [==============================] - 0s 2ms/step - loss: 0.7004 - acc:
0.6615
Epoch 15/100
18/18 [==============================] - 0s 6ms/step - loss: 0.6876 - acc:
0.6615
Epoch 16/100
18/18 [==============================] - 0s 6ms/step - loss: 0.6832 - acc:
0.6597
Epoch 17/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6782 - acc:
0.6597
Epoch 18/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6767 - acc:
0.6597
Epoch 19/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6725 - acc:
0.6615
Epoch 20/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6753 - acc:
0.6580
Epoch 21/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6666 - acc:
0.6615
Epoch 22/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6649 - acc:
0.6615
Epoch 23/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6666 - acc:
0.6597
```

```
Epoch 24/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6620 - acc:
0.6615
Epoch 25/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6609 - acc:
0.6597
Epoch 26/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6609 - acc:
0.6615
Epoch 27/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6571 - acc:
0.6615
Epoch 28/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6587 - acc:
0.6597
Epoch 29/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6571 - acc:
0.6597
Epoch 30/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6549 - acc:
0.6615
Epoch 31/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6541 - acc:
0.6615
Epoch 32/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6576 - acc:
0.6580
Epoch 33/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6553 - acc:
0.6615
Epoch 34/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6535 - acc:
0.6597
Epoch 35/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6503 - acc:
0.6615
Epoch 36/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6513 - acc:
0.6615
Epoch 37/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6507 - acc:
0.6615
Epoch 38/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6476 - acc:
0.6615
Epoch 39/100
```

```
18/18 [==============================] - 0s 3ms/step - loss: 0.6481 - acc:
0.6615
Epoch 40/100
18/18 [==============================] - 0s 5ms/step - loss: 0.6464 - acc:
0.6615
Epoch 41/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6460 - acc:
0.6615
Epoch 42/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6457 - acc:
0.6615
Epoch 43/100
18/18 [==============================] - 0s 5ms/step - loss: 0.6444 - acc:
0.6632
Epoch 44/100
18/18 [==============================] - 0s 5ms/step - loss: 0.6463 - acc:
0.6615
Epoch 45/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6434 - acc:
0.6632
Epoch 46/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6437 - acc:
0.6615
Epoch 47/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6433 - acc:
0.6615
Epoch 48/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6450 - acc:
0.6615
Epoch 49/100
18/18 [==============================] - 0s 5ms/step - loss: 0.6457 - acc:
0.6597
Epoch 50/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6436 - acc:
0.6597
Epoch 51/100
18/18 [==============================] - 0s 5ms/step - loss: 0.6413 - acc:
0.6632
Epoch 52/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6419 - acc:
0.6632
Epoch 53/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6407 - acc:
0.6632
Epoch 54/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6420 - acc:
0.6615
```

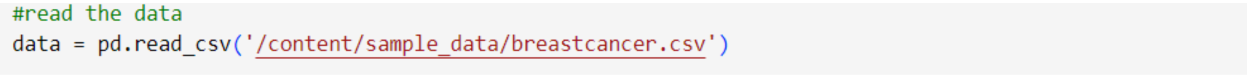
```
Epoch 55/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6411 - acc:
0.6615
Epoch 56/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6399 - acc:
0.6632
Epoch 57/100
18/18 [==============================] - 0s 5ms/step - loss: 0.6404 - acc:
0.6615
Epoch 58/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6401 - acc:
0.6632
Epoch 59/100
18/18 [==============================] - 0s 5ms/step - loss: 0.6392 - acc:
0.6632
Epoch 60/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6395 - acc:
0.6632
Epoch 61/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6390 - acc:
0.6632
Epoch 62/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6390 - acc:
0.6632
Epoch 63/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6405 - acc:
0.6615
Epoch 64/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6392 - acc:
0.6615
Epoch 65/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6389 - acc:
0.6632
Epoch 66/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6394 - acc:
0.6615
Epoch 67/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6382 - acc:
0.6632
Epoch 68/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6381 - acc:
0.6632
Epoch 69/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6378 - acc:
0.6632
Epoch 70/100
```

```
18/18 [==============================] - 0s 2ms/step - loss: 0.6377 - acc:
0.6632
Epoch 71/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6380 - acc:
0.6632
Epoch 72/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6383 - acc:
0.6632
Epoch 73/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6387 - acc:
0.6615
Epoch 74/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6371 - acc:
0.6632
Epoch 75/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6385 - acc:
0.6632
Epoch 76/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6409 - acc:
0.6615
Epoch 77/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6369 - acc:
0.6632
Epoch 78/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6389 - acc:
0.6615
Epoch 79/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6370 - acc:
0.6632
Epoch 80/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6375 - acc:
0.6632
Epoch 81/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6372 - acc:
0.6632
Epoch 82/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6371 - acc:
0.6632
Epoch 83/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6367 - acc:
0.6632
Epoch 84/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6367 - acc:
0.6632
Epoch 85/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6368 - acc:
0.6632
```

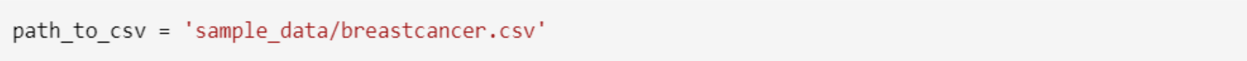
```
Epoch 86/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6369 - acc:
0.6632
Epoch 87/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6364 - acc:
0.6632
Epoch 88/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6372 - acc:
0.6632
Epoch 89/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6368 - acc:
0.6632
Epoch 90/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6368 - acc:
0.6632
Epoch 91/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6368 - acc:
0.6632
Epoch 92/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6367 - acc:
0.6632
Epoch 93/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6366 - acc:
0.6632
Epoch 94/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6368 - acc:
0.6632
Epoch 95/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6367 - acc:
0.6632
Epoch 96/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6372 - acc:
0.6632
Epoch 97/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6373 - acc:
0.6632
Epoch 98/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6364 - acc:
0.6632
Epoch 99/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6364 - acc:
0.6632
Epoch 100/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6362 - acc:
0.6632
Model: "sequential"
_________________________________________________________________
```

```
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 20)                180

 dense_1 (Dense)             (None, 4)                 84

 dense_2 (Dense)             (None, 1)                 5

=================================================================
Total params: 269 (1.05 KB)
Trainable params: 269 (1.05 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
6/6 [==============================] - 0s 3ms/step - loss: 0.6607 - acc:
0.6250
[0.660732090473175, 0.625]
```

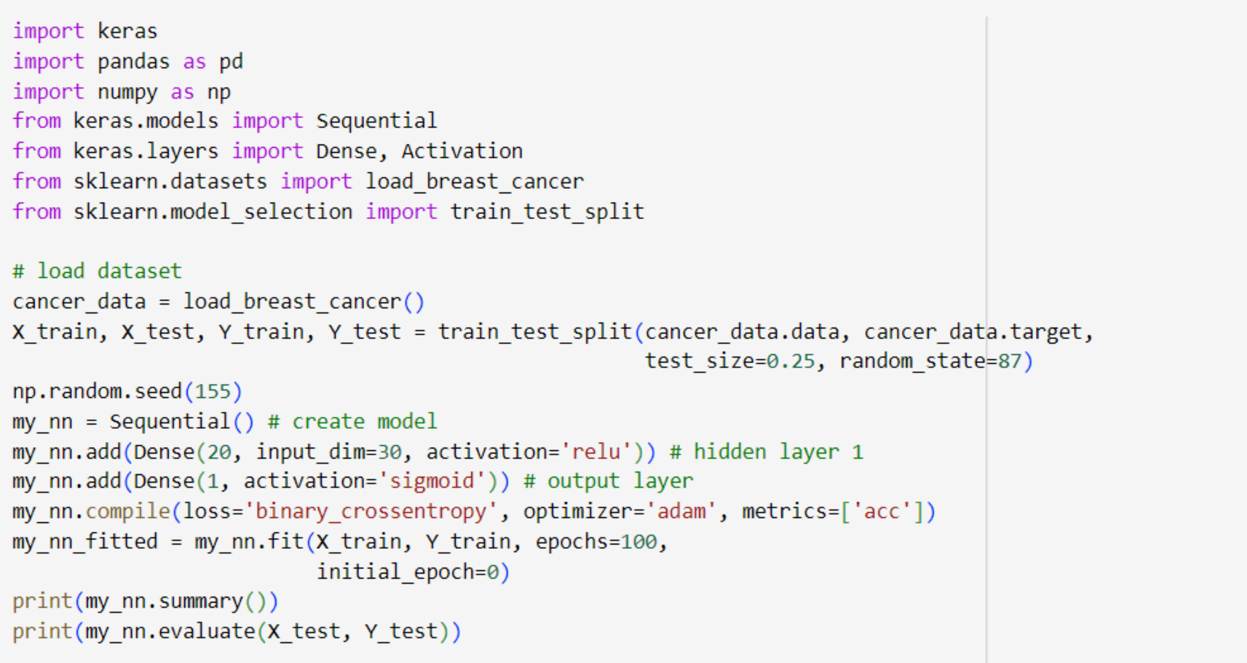
CODE:



```
#read the data
data = pd.read_csv('/content/sample_data/breastcancer.csv')
```



```
path_to_csv = 'sample_data/breastcancer.csv'
```



```
import keras
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split

# load dataset
cancer_data = load_breast_cancer()
X_train, X_test, Y_train, Y_test = train_test_split(cancer_data.data, cancer_data.target,
                                                    test_size=0.25, random_state=87)
np.random.seed(155)
my_nn = Sequential() # create model
my_nn.add(Dense(20, input_dim=30, activation='relu')) # hidden layer 1
my_nn.add(Dense(1, activation='sigmoid')) # output layer
my_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
my_nn_fitted = my_nn.fit(X_train, Y_train, epochs=100,
                         initial_epoch=0)
print(my_nn.summary())
print(my_nn.evaluate(X_test, Y_test))
```

OUTPUT:

```
Epoch 1/100
```

```
14/14 [==============================] - 1s 2ms/step - loss: 123.0207 -
acc: 0.6197
Epoch 2/100
14/14 [==============================] - 0s 2ms/step - loss: 60.4747 -
acc: 0.6197
Epoch 3/100
14/14 [==============================] - 0s 2ms/step - loss: 9.4681 - acc:
0.6667
Epoch 4/100
14/14 [==============================] - 0s 2ms/step - loss: 3.8449 - acc:
0.5962
Epoch 5/100
14/14 [==============================] - 0s 3ms/step - loss: 1.4987 - acc:
0.9061
Epoch 6/100
14/14 [==============================] - 0s 3ms/step - loss: 1.4846 - acc:
0.8991
Epoch 7/100
14/14 [==============================] - 0s 3ms/step - loss: 1.3028 - acc:
0.8474
Epoch 8/100
14/14 [==============================] - 0s 3ms/step - loss: 1.2434 - acc:
0.8991
Epoch 9/100
14/14 [==============================] - 0s 3ms/step - loss: 1.2105 - acc:
0.8991
Epoch 10/100
14/14 [==============================] - 0s 3ms/step - loss: 1.1615 - acc:
0.8779
Epoch 11/100
14/14 [==============================] - 0s 3ms/step - loss: 1.1214 - acc:
0.8897
Epoch 12/100
14/14 [==============================] - 0s 4ms/step - loss: 1.0958 - acc:
0.8850
Epoch 13/100
14/14 [==============================] - 0s 3ms/step - loss: 1.0861 - acc:
0.8920
Epoch 14/100
14/14 [==============================] - 0s 3ms/step - loss: 1.0668 - acc:
0.8826
Epoch 15/100
14/14 [==============================] - 0s 3ms/step - loss: 1.0333 - acc:
0.9038
Epoch 16/100
14/14 [==============================] - 0s 4ms/step - loss: 1.0108 - acc:
0.8920
```

```
Epoch 17/100
14/14 [==============================] - 0s 3ms/step - loss: 1.0230 - acc:
0.9085
Epoch 18/100
14/14 [==============================] - 0s 4ms/step - loss: 1.0267 - acc:
0.8850
Epoch 19/100
14/14 [==============================] - 0s 3ms/step - loss: 0.9551 - acc:
0.9038
Epoch 20/100
14/14 [==============================] - 0s 3ms/step - loss: 0.9282 - acc:
0.9014
Epoch 21/100
14/14 [==============================] - 0s 3ms/step - loss: 0.9307 - acc:
0.9202
Epoch 22/100
14/14 [==============================] - 0s 4ms/step - loss: 0.8851 - acc:
0.9085
Epoch 23/100
14/14 [==============================] - 0s 4ms/step - loss: 0.8757 - acc:
0.9108
Epoch 24/100
14/14 [==============================] - 0s 4ms/step - loss: 0.8525 - acc:
0.9108
Epoch 25/100
14/14 [==============================] - 0s 4ms/step - loss: 0.8538 - acc:
0.9249
Epoch 26/100
14/14 [==============================] - 0s 3ms/step - loss: 0.8148 - acc:
0.9061
Epoch 27/100
14/14 [==============================] - 0s 4ms/step - loss: 0.8203 - acc:
0.9225
Epoch 28/100
14/14 [==============================] - 0s 4ms/step - loss: 0.8315 - acc:
0.8920
Epoch 29/100
14/14 [==============================] - 0s 4ms/step - loss: 0.8221 - acc:
0.9131
Epoch 30/100
14/14 [==============================] - 0s 5ms/step - loss: 0.7547 - acc:
0.9131
Epoch 31/100
14/14 [==============================] - 0s 4ms/step - loss: 0.7553 - acc:
0.9038
Epoch 32/100
```

```
14/14 [==============================] - 0s 5ms/step - loss: 0.8140 - acc:
0.9225
Epoch 33/100
14/14 [==============================] - 0s 3ms/step - loss: 0.8189 - acc:
0.8920
Epoch 34/100
14/14 [==============================] - 0s 4ms/step - loss: 0.7379 - acc:
0.9249
Epoch 35/100
14/14 [==============================] - 0s 4ms/step - loss: 0.7128 - acc:
0.9061
Epoch 36/100
14/14 [==============================] - 0s 4ms/step - loss: 0.7098 - acc:
0.9249
Epoch 37/100
14/14 [==============================] - 0s 5ms/step - loss: 0.6937 - acc:
0.9155
Epoch 38/100
14/14 [==============================] - 0s 4ms/step - loss: 0.6956 - acc:
0.9014
Epoch 39/100
14/14 [==============================] - 0s 4ms/step - loss: 0.6955 - acc:
0.9272
Epoch 40/100
14/14 [==============================] - 0s 4ms/step - loss: 0.6942 - acc:
0.9014
Epoch 41/100
14/14 [==============================] - 0s 6ms/step - loss: 0.6627 - acc:
0.9155
Epoch 42/100
14/14 [==============================] - 0s 5ms/step - loss: 0.6556 - acc:
0.9155
Epoch 43/100
14/14 [==============================] - 0s 5ms/step - loss: 0.6548 - acc:
0.9249
Epoch 44/100
14/14 [==============================] - 0s 5ms/step - loss: 0.6478 - acc:
0.9202
Epoch 45/100
14/14 [==============================] - 0s 4ms/step - loss: 0.6364 - acc:
0.9178
Epoch 46/100
14/14 [==============================] - 0s 4ms/step - loss: 0.6463 - acc:
0.9225
Epoch 47/100
14/14 [==============================] - 0s 3ms/step - loss: 0.6613 - acc:
0.9155
```

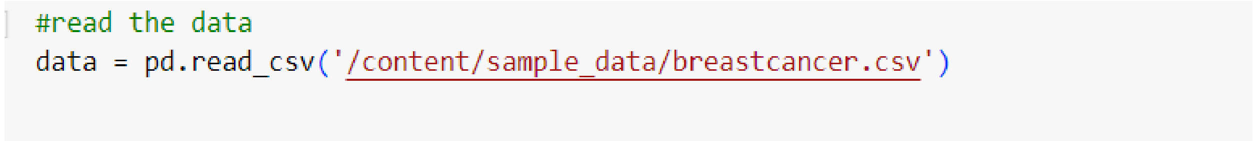
```
Epoch 48/100
14/14 [==============================] - 0s 5ms/step - loss: 0.6329 - acc:
0.9272
Epoch 49/100
14/14 [==============================] - 0s 4ms/step - loss: 0.6271 - acc:
0.9202
Epoch 50/100
14/14 [==============================] - 0s 4ms/step - loss: 0.6229 - acc:
0.9225
Epoch 51/100
14/14 [==============================] - 0s 3ms/step - loss: 0.6452 - acc:
0.9249
Epoch 52/100
14/14 [==============================] - 0s 5ms/step - loss: 0.7427 - acc:
0.8944
Epoch 53/100
14/14 [==============================] - 0s 6ms/step - loss: 0.6845 - acc:
0.9296
Epoch 54/100
14/14 [==============================] - 0s 5ms/step - loss: 0.5932 - acc:
0.9155
Epoch 55/100
14/14 [==============================] - 0s 4ms/step - loss: 0.6326 - acc:
0.9225
Epoch 56/100
14/14 [==============================] - 0s 5ms/step - loss: 0.6030 - acc:
0.9038
Epoch 57/100
14/14 [==============================] - 0s 5ms/step - loss: 0.5981 - acc:
0.9249
Epoch 58/100
14/14 [==============================] - 0s 4ms/step - loss: 0.5798 - acc:
0.9225
Epoch 59/100
14/14 [==============================] - 0s 4ms/step - loss: 0.5942 - acc:
0.9202
Epoch 60/100
14/14 [==============================] - 0s 5ms/step - loss: 0.5803 - acc:
0.9249
Epoch 61/100
14/14 [==============================] - 0s 4ms/step - loss: 0.5751 - acc:
0.9178
Epoch 62/100
14/14 [==============================] - 0s 3ms/step - loss: 0.5694 - acc:
0.9155
Epoch 63/100
```

```
14/14 [==============================] - 0s 3ms/step - loss: 0.5892 - acc:
0.9272
Epoch 64/100
14/14 [==============================] - 0s 3ms/step - loss: 0.5811 - acc:
0.9155
Epoch 65/100
14/14 [==============================] - 0s 3ms/step - loss: 0.5729 - acc:
0.9319
Epoch 66/100
14/14 [==============================] - 0s 3ms/step - loss: 0.6445 - acc:
0.9085
Epoch 67/100
14/14 [==============================] - 0s 3ms/step - loss: 0.6177 - acc:
0.9108
Epoch 68/100
14/14 [==============================] - 0s 3ms/step - loss: 0.6348 - acc:
0.9272
Epoch 69/100
14/14 [==============================] - 0s 3ms/step - loss: 0.6680 - acc:
0.8991
Epoch 70/100
14/14 [==============================] - 0s 3ms/step - loss: 0.7321 - acc:
0.9225
Epoch 71/100
14/14 [==============================] - 0s 3ms/step - loss: 0.6012 - acc:
0.9061
Epoch 72/100
14/14 [==============================] - 0s 3ms/step - loss: 0.5508 - acc:
0.9272
Epoch 73/100
14/14 [==============================] - 0s 3ms/step - loss: 0.5905 - acc:
0.9178
Epoch 74/100
14/14 [==============================] - 0s 3ms/step - loss: 0.5992 - acc:
0.9108
Epoch 75/100
14/14 [==============================] - 0s 3ms/step - loss: 0.6175 - acc:
0.9178
Epoch 76/100
14/14 [==============================] - 0s 3ms/step - loss: 0.5192 - acc:
0.9108
Epoch 77/100
14/14 [==============================] - 0s 3ms/step - loss: 0.6331 - acc:
0.9249
Epoch 78/100
14/14 [==============================] - 0s 3ms/step - loss: 0.5411 - acc:
0.9319
```

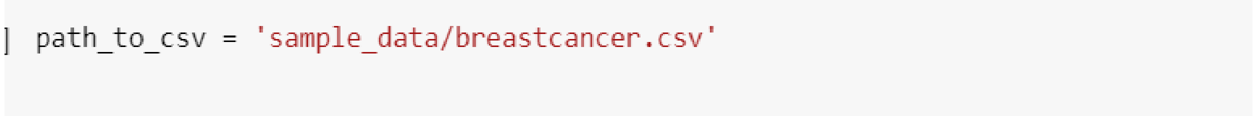
```
Epoch 79/100
14/14 [==============================] - 0s 3ms/step - loss: 0.5035 - acc:
0.9272
Epoch 80/100
14/14 [==============================] - 0s 2ms/step - loss: 0.5120 - acc:
0.9296
Epoch 81/100
14/14 [==============================] - 0s 3ms/step - loss: 0.6046 - acc:
0.9202
Epoch 82/100
14/14 [==============================] - 0s 3ms/step - loss: 0.5400 - acc:
0.9202
Epoch 83/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4835 - acc:
0.9296
Epoch 84/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4973 - acc:
0.9178
Epoch 85/100
14/14 [==============================] - 0s 2ms/step - loss: 0.5025 - acc:
0.9296
Epoch 86/100
14/14 [==============================] - 0s 3ms/step - loss: 0.5157 - acc:
0.9155
Epoch 87/100
14/14 [==============================] - 0s 3ms/step - loss: 0.5045 - acc:
0.9249
Epoch 88/100
14/14 [==============================] - 0s 3ms/step - loss: 0.5077 - acc:
0.9343
Epoch 89/100
14/14 [==============================] - 0s 2ms/step - loss: 0.5645 - acc:
0.8967
Epoch 90/100
14/14 [==============================] - 0s 3ms/step - loss: 0.5201 - acc:
0.9202
Epoch 91/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4507 - acc:
0.9296
Epoch 92/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4538 - acc:
0.9296
Epoch 93/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4506 - acc:
0.9225
Epoch 94/100
```

```
14/14 [==============================] - 0s 3ms/step - loss: 0.4651 - acc:
0.9202
Epoch 95/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4839 - acc:
0.9319
Epoch 96/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4979 - acc:
0.9061
Epoch 97/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4415 - acc:
0.9272
Epoch 98/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4435 - acc:
0.9225
Epoch 99/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4459 - acc:
0.9131
Epoch 100/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4462 - acc:
0.9272
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 20)                620

 dense_4 (Dense)             (None, 1)                 21

=================================================================
Total params: 641 (2.50 KB)
Trainable params: 641 (2.50 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
5/5 [==============================] - 0s 4ms/step - loss: 1.3253 - acc:
0.8392
[1.3253005743026733, 0.8391608595848083]
```

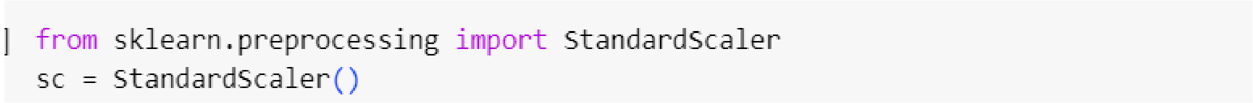
CODE:



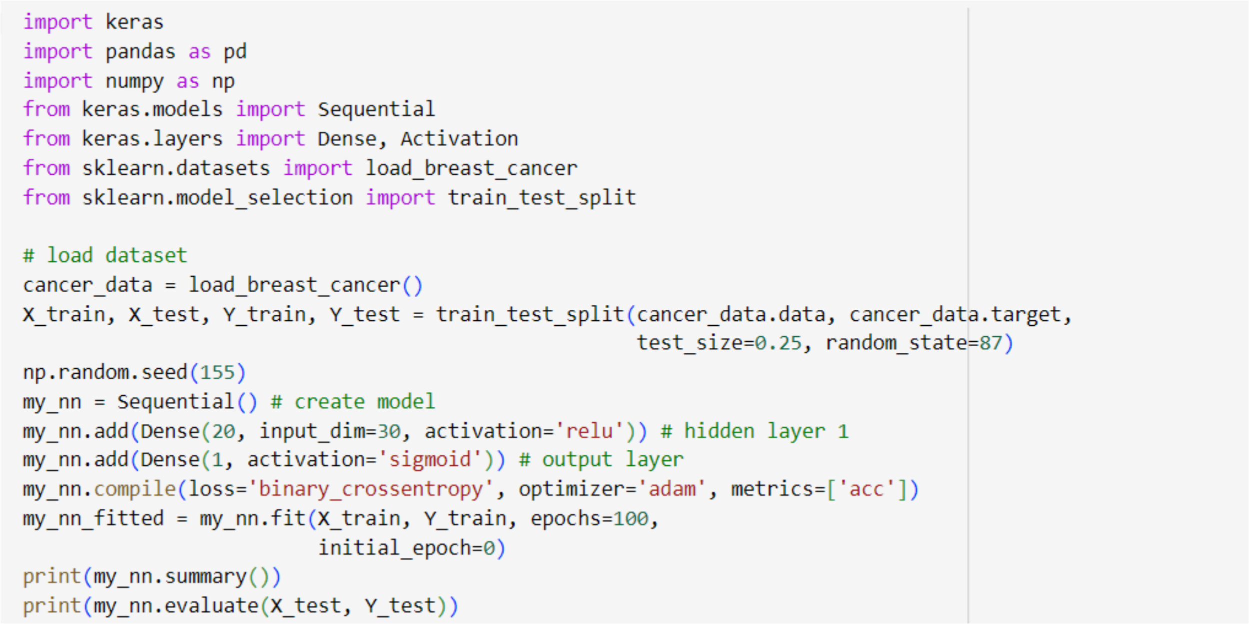
```
] #read the data
  data = pd.read_csv('/content/sample_data/breastcancer.csv')
```



```
] path_to_csv = 'sample_data/breastcancer.csv'
```



```
] from sklearn.preprocessing import StandardScaler
  sc = StandardScaler()
```



```
import keras
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split

# load dataset
cancer_data = load_breast_cancer()
X_train, X_test, Y_train, Y_test = train_test_split(cancer_data.data, cancer_data.target,
                                                     test_size=0.25, random_state=87)
np.random.seed(155)
my_nn = Sequential() # create model
my_nn.add(Dense(20, input_dim=30, activation='relu')) # hidden layer 1
my_nn.add(Dense(1, activation='sigmoid')) # output layer
my_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
my_nn_fitted = my_nn.fit(X_train, Y_train, epochs=100,
                         initial_epoch=0)
print(my_nn.summary())
print(my_nn.evaluate(X_test, Y_test))
```

OUTPUT:

```
Epoch 1/100
14/14 [==============================] - 1s 2ms/step - loss: 43.0876 - acc: 0.3803
Epoch 2/100
14/14 [==============================] - 0s 2ms/step - loss: 12.1806 - acc: 0.4437
Epoch 3/100
14/14 [==============================] - 0s 3ms/step - loss: 6.8787 - acc: 0.6174
Epoch 4/100
14/14 [==============================] - 0s 3ms/step - loss: 4.5784 - acc: 0.5376
Epoch 5/100
14/14 [==============================] - 0s 3ms/step - loss: 3.7497 - acc: 0.6174
Epoch 6/100
14/14 [==============================] - 0s 2ms/step - loss: 3.1469 - acc: 0.6221
Epoch 7/100
14/14 [==============================] - 0s 2ms/step - loss: 2.7620 - acc: 0.6479
Epoch 8/100
```

```
14/14 [==============================] - 0s 2ms/step - loss: 2.5535 - acc: 0.7042
Epoch 9/100
14/14 [==============================] - 0s 2ms/step - loss: 2.2863 - acc: 0.7042
Epoch 10/100
14/14 [==============================] - 0s 2ms/step - loss: 2.0869 - acc: 0.7300
Epoch 11/100
14/14 [==============================] - 0s 3ms/step - loss: 1.9328 - acc: 0.7277
Epoch 12/100
14/14 [==============================] - 0s 2ms/step - loss: 1.8578 - acc: 0.7371
Epoch 13/100
14/14 [==============================] - 0s 2ms/step - loss: 1.7023 - acc: 0.7582
Epoch 14/100
14/14 [==============================] - 0s 2ms/step - loss: 1.6321 - acc: 0.7582
Epoch 15/100
14/14 [==============================] - 0s 2ms/step - loss: 1.5860 - acc: 0.7864
Epoch 16/100
14/14 [==============================] - 0s 2ms/step - loss: 1.4604 - acc: 0.7934
Epoch 17/100
14/14 [==============================] - 0s 4ms/step - loss: 1.4102 - acc: 0.7864
Epoch 18/100
14/14 [==============================] - 0s 6ms/step - loss: 1.3337 - acc: 0.8099
Epoch 19/100
14/14 [==============================] - 0s 12ms/step - loss: 1.2667 - acc: 0.7887
Epoch 20/100
14/14 [==============================] - 0s 7ms/step - loss: 1.2334 - acc: 0.8239
Epoch 21/100
14/14 [==============================] - 0s 5ms/step - loss: 1.1966 - acc: 0.8005
Epoch 22/100
14/14 [==============================] - 0s 3ms/step - loss: 1.1164 - acc: 0.8263
Epoch 23/100
14/14 [==============================] - 0s 3ms/step - loss: 1.0899 - acc: 0.8052
Epoch 24/100
14/14 [==============================] - 0s 3ms/step - loss: 1.0664 - acc: 0.8169
Epoch 25/100
14/14 [==============================] - 0s 3ms/step - loss: 1.0696 - acc: 0.8310
Epoch 26/100
14/14 [==============================] - 0s 3ms/step - loss: 1.0683 - acc: 0.8333
Epoch 27/100
14/14 [==============================] - 0s 4ms/step - loss: 0.9202 - acc: 0.8146
Epoch 28/100
14/14 [==============================] - 0s 3ms/step - loss: 0.9490 - acc: 0.8521
Epoch 29/100
14/14 [==============================] - 0s 4ms/step - loss: 0.8645 - acc: 0.8333
Epoch 30/100
14/14 [==============================] - 0s 4ms/step - loss: 0.8390 - acc: 0.8380
Epoch 31/100
14/14 [==============================] - 0s 3ms/step - loss: 0.8051 - acc: 0.8709
```

```
Epoch 32/100
14/14 [==============================] - 0s 3ms/step - loss: 0.8159 - acc: 0.8380
Epoch 33/100
14/14 [==============================] - 0s 3ms/step - loss: 0.8131 - acc: 0.8568
Epoch 34/100
14/14 [==============================] - 0s 3ms/step - loss: 0.7494 - acc: 0.8685
Epoch 35/100
14/14 [==============================] - 0s 3ms/step - loss: 0.6955 - acc: 0.8592
Epoch 36/100
14/14 [==============================] - 0s 3ms/step - loss: 0.7510 - acc: 0.8451
Epoch 37/100
14/14 [==============================] - 0s 3ms/step - loss: 0.7873 - acc: 0.8427
Epoch 38/100
14/14 [==============================] - 0s 3ms/step - loss: 0.6854 - acc: 0.8920
Epoch 39/100
14/14 [==============================] - 0s 3ms/step - loss: 0.5884 - acc: 0.8709
Epoch 40/100
14/14 [==============================] - 0s 3ms/step - loss: 0.5930 - acc: 0.8873
Epoch 41/100
14/14 [==============================] - 0s 4ms/step - loss: 0.5530 - acc: 0.8685
Epoch 42/100
14/14 [==============================] - 0s 4ms/step - loss: 0.5901 - acc: 0.8897
Epoch 43/100
14/14 [==============================] - 0s 3ms/step - loss: 0.5046 - acc: 0.8897
Epoch 44/100
14/14 [==============================] - 0s 3ms/step - loss: 0.5049 - acc: 0.8850
Epoch 45/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4889 - acc: 0.8873
Epoch 46/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4905 - acc: 0.8779
Epoch 47/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4676 - acc: 0.8897
Epoch 48/100
14/14 [==============================] - 0s 4ms/step - loss: 0.4491 - acc: 0.8967
Epoch 49/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4442 - acc: 0.8991
Epoch 50/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4267 - acc: 0.8873
Epoch 51/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4494 - acc: 0.8967
Epoch 52/100
14/14 [==============================] - 0s 3ms/step - loss: 0.3913 - acc: 0.8967
Epoch 53/100
14/14 [==============================] - 0s 3ms/step - loss: 0.3841 - acc: 0.8944
Epoch 54/100
14/14 [==============================] - 0s 3ms/step - loss: 0.3747 - acc: 0.9038
Epoch 55/100
```

```
14/14 [==============================] - 0s 3ms/step - loss: 0.3860 - acc: 0.9155
Epoch 56/100
14/14 [==============================] - 0s 3ms/step - loss: 0.3531 - acc: 0.9014
Epoch 57/100
14/14 [==============================] - 0s 3ms/step - loss: 0.3438 - acc: 0.9108
Epoch 58/100
14/14 [==============================] - 0s 3ms/step - loss: 0.3605 - acc: 0.9014
Epoch 59/100
14/14 [==============================] - 0s 5ms/step - loss: 0.3394 - acc: 0.9038
Epoch 60/100
14/14 [==============================] - 0s 5ms/step - loss: 0.3281 - acc: 0.9038
Epoch 61/100
14/14 [==============================] - 0s 3ms/step - loss: 0.3383 - acc: 0.9061
Epoch 62/100
14/14 [==============================] - 0s 5ms/step - loss: 0.4140 - acc: 0.9085
Epoch 63/100
14/14 [==============================] - 0s 3ms/step - loss: 0.2953 - acc: 0.9131
Epoch 64/100
14/14 [==============================] - 0s 3ms/step - loss: 0.3400 - acc: 0.9108
Epoch 65/100
14/14 [==============================] - 0s 4ms/step - loss: 0.3323 - acc: 0.9085
Epoch 66/100
14/14 [==============================] - 0s 5ms/step - loss: 0.2689 - acc: 0.9155
Epoch 67/100
14/14 [==============================] - 0s 4ms/step - loss: 0.2539 - acc: 0.9249
Epoch 68/100
14/14 [==============================] - 0s 4ms/step - loss: 0.3391 - acc: 0.8991
Epoch 69/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4339 - acc: 0.8779
Epoch 70/100
14/14 [==============================] - 0s 3ms/step - loss: 0.2653 - acc: 0.9178
Epoch 71/100
14/14 [==============================] - 0s 3ms/step - loss: 0.2398 - acc: 0.9178
Epoch 72/100
14/14 [==============================] - 0s 5ms/step - loss: 0.2523 - acc: 0.9131
Epoch 73/100
14/14 [==============================] - 0s 5ms/step - loss: 0.2456 - acc: 0.9272
Epoch 74/100
14/14 [==============================] - 0s 3ms/step - loss: 0.2269 - acc: 0.9319
Epoch 75/100
14/14 [==============================] - 0s 3ms/step - loss: 0.2320 - acc: 0.9319
Epoch 76/100
14/14 [==============================] - 0s 4ms/step - loss: 0.2300 - acc: 0.9225
Epoch 77/100
14/14 [==============================] - 0s 5ms/step - loss: 0.2072 - acc: 0.9272
Epoch 78/100
14/14 [==============================] - 0s 5ms/step - loss: 0.3361 - acc: 0.8991
```

```
Epoch 79/100
14/14 [==============================] - 0s 3ms/step - loss: 0.2428 - acc: 0.9225
Epoch 80/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2091 - acc: 0.9272
Epoch 81/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1938 - acc: 0.9319
Epoch 82/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2027 - acc: 0.9366
Epoch 83/100
14/14 [==============================] - 0s 3ms/step - loss: 0.2558 - acc: 0.9155
Epoch 84/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1852 - acc: 0.9319
Epoch 85/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2217 - acc: 0.9202
Epoch 86/100
14/14 [==============================] - 0s 3ms/step - loss: 0.2690 - acc: 0.9178
Epoch 87/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1848 - acc: 0.9319
Epoch 88/100
14/14 [==============================] - 0s 3ms/step - loss: 0.1996 - acc: 0.9296
Epoch 89/100
14/14 [==============================] - 0s 3ms/step - loss: 0.1726 - acc: 0.9343
Epoch 90/100
14/14 [==============================] - 0s 3ms/step - loss: 0.1866 - acc: 0.9413
Epoch 91/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1746 - acc: 0.9319
Epoch 92/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2494 - acc: 0.9225
Epoch 93/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2672 - acc: 0.9108
Epoch 94/100
14/14 [==============================] - 0s 3ms/step - loss: 0.1618 - acc: 0.9413
Epoch 95/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1757 - acc: 0.9390
Epoch 96/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1665 - acc: 0.9343
Epoch 97/100
14/14 [==============================] - 0s 3ms/step - loss: 0.1665 - acc: 0.9413
Epoch 98/100
14/14 [==============================] - 0s 3ms/step - loss: 0.1696 - acc: 0.9413
Epoch 99/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2100 - acc: 0.9202
Epoch 100/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1602 - acc: 0.9390
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
```

```
=================================================================
 dense_5 (Dense)          (None, 20)            620

 dense_6 (Dense)          (None, 1)            21

=================================================================
Total params: 641 (2.50 KB)
Trainable params: 641 (2.50 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
5/5 [==============================] - 0s 3ms/step - loss: 0.2860 - acc: 0.8741
[0.2860058546066284, 0.8741258978843689]
```

addCode
addText

CODE:

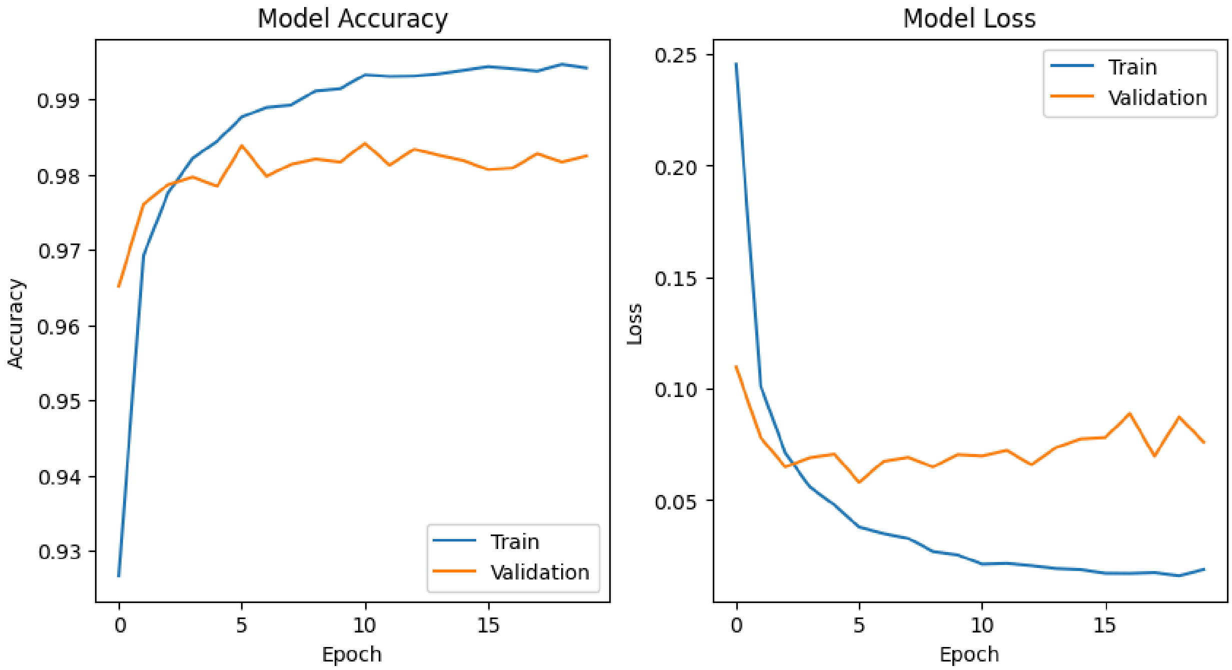
```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt

# load MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# normalize pixel values to range [0, 1]
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255

# convert class labels to binary class matrices
num_classes = 10
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

# create a simple neural network model
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

# train the model and record the training history
history = model.fit(x_train.reshape(-1, 784), y_train,
validation_data=(x_test.reshape(-1, 784), y_test),
                    epochs=20, batch_size=128)

# plot the training and validation accuracy and loss curves
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='lower right')

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper right')

plt.show()
```

OUTPUT:

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
Epoch 1/20
469/469 [==============================] - 13s 24ms/step - loss: 0.2454 - accuracy:
0.9267 - val_loss: 0.1097 - val_accuracy: 0.9652
Epoch 2/20
469/469 [==============================] - 11s 24ms/step - loss: 0.1012 - accuracy:
0.9693 - val_loss: 0.0780 - val_accuracy: 0.9761
Epoch 3/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0711 - accuracy:
0.9776 - val_loss: 0.0648 - val_accuracy: 0.9787
Epoch 4/20
```

```
469/469 [==============================] - 11s 24ms/step - loss: 0.0558 - accuracy:
0.9822 - val_loss: 0.0690 - val_accuracy: 0.9797
Epoch 5/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0477 - accuracy:
0.9845 - val_loss: 0.0706 - val_accuracy: 0.9785
Epoch 6/20
469/469 [==============================] - 11s 22ms/step - loss: 0.0378 - accuracy:
0.9877 - val_loss: 0.0580 - val_accuracy: 0.9839
Epoch 7/20
469/469 [==============================] - 14s 30ms/step - loss: 0.0349 - accuracy:
0.9890 - val_loss: 0.0674 - val_accuracy: 0.9798
Epoch 8/20
469/469 [==============================] - 14s 30ms/step - loss: 0.0328 - accuracy:
0.9893 - val_loss: 0.0691 - val_accuracy: 0.9814
Epoch 9/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0269 - accuracy:
0.9912 - val_loss: 0.0648 - val_accuracy: 0.9821
Epoch 10/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0254 - accuracy:
0.9915 - val_loss: 0.0703 - val_accuracy: 0.9817
Epoch 11/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0212 - accuracy:
0.9933 - val_loss: 0.0697 - val_accuracy: 0.9842
Epoch 12/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0216 - accuracy:
0.9931 - val_loss: 0.0722 - val_accuracy: 0.9813
Epoch 13/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0205 - accuracy:
0.9931 - val_loss: 0.0658 - val_accuracy: 0.9834
Epoch 14/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0193 - accuracy:
0.9934 - val_loss: 0.0735 - val_accuracy: 0.9826
Epoch 15/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0188 - accuracy:
0.9939 - val_loss: 0.0774 - val_accuracy: 0.9819
Epoch 16/20
469/469 [==============================] - 10s 22ms/step - loss: 0.0172 - accuracy:
0.9944 - val_loss: 0.0780 - val_accuracy: 0.9807
Epoch 17/20
469/469 [==============================] - 12s 26ms/step - loss: 0.0172 - accuracy:
0.9941 - val_loss: 0.0887 - val_accuracy: 0.9809
Epoch 18/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0175 - accuracy:
0.9938 - val_loss: 0.0697 - val_accuracy: 0.9828
```

```
Epoch 19/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0161 - accuracy:
0.9947 - val_loss: 0.0873 - val_accuracy: 0.9817
Epoch 20/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0189 - accuracy:
0.9942 - val_loss: 0.0760 - val_accuracy: 0.9825
```



CODE:

```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

# load MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# normalize pixel values to range [0, 1]
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255

# convert class labels to binary class matrices
num_classes = 10
```

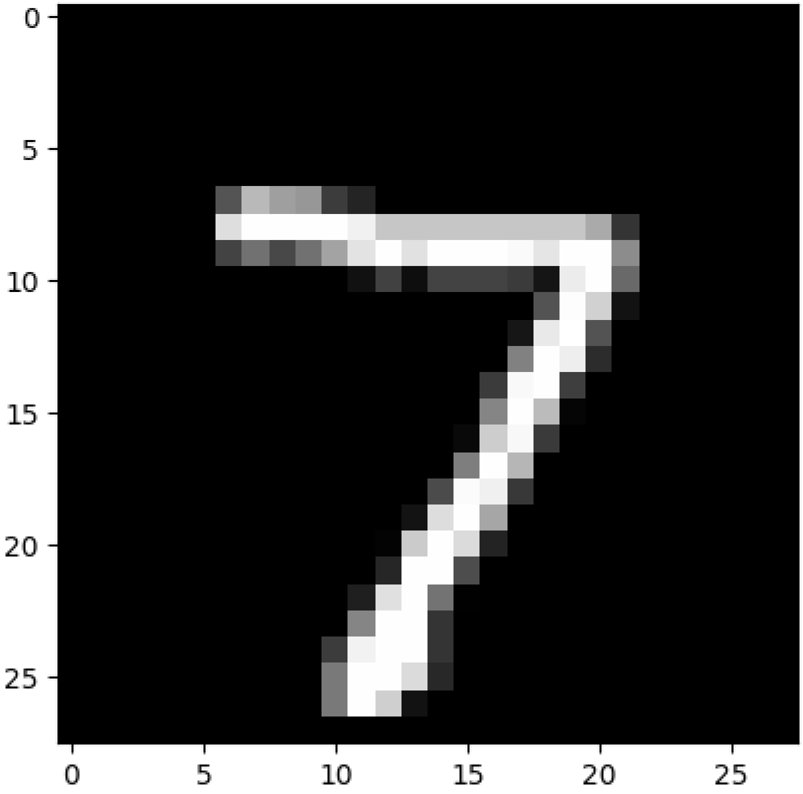
```
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

# create a simple neural network model
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))

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

# train the model
model.fit(x_train.reshape(-1, 784), y_train,
validation_data=(x_test.reshape(-1, 784), y_test),
          epochs=20, batch_size=128)

# plot one of the images in the test data
plt.imshow(x_test[0], cmap='gray')
plt.show()

# make a prediction on the image using the trained model
prediction = model.predict(x_test[0].reshape(1, -1))
print('Model prediction:', np.argmax(prediction))
```

OUTPUT:

```
Epoch 1/20
469/469 [==============================] - 14s 26ms/step - loss: 0.2496 - accuracy:
0.9246 - val_loss: 0.1082 - val_accuracy: 0.9654
Epoch 2/20
469/469 [==============================] - 11s 24ms/step - loss: 0.1016 - accuracy:
0.9687 - val_loss: 0.0789 - val_accuracy: 0.9735
Epoch 3/20
469/469 [==============================] - 9s 20ms/step - loss: 0.0696 - accuracy: 0.9784
- val_loss: 0.0724 - val_accuracy: 0.9766
Epoch 4/20
469/469 [==============================] - 10s 22ms/step - loss: 0.0530 - accuracy:
0.9832 - val_loss: 0.0670 - val_accuracy: 0.9795
Epoch 5/20
```

```
469/469 [==============================] - 11s 24ms/step - loss: 0.0448 - accuracy:
0.9854 - val_loss: 0.0595 - val_accuracy: 0.9818
Epoch 6/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0395 - accuracy:
0.9870 - val_loss: 0.0602 - val_accuracy: 0.9820
Epoch 7/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0338 - accuracy:
0.9891 - val_loss: 0.0677 - val_accuracy: 0.9807
Epoch 8/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0294 - accuracy:
0.9899 - val_loss: 0.0614 - val_accuracy: 0.9825
Epoch 9/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0267 - accuracy:
0.9911 - val_loss: 0.0658 - val_accuracy: 0.9822
Epoch 10/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0254 - accuracy:
0.9912 - val_loss: 0.0701 - val_accuracy: 0.9820
Epoch 11/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0256 - accuracy:
0.9916 - val_loss: 0.0694 - val_accuracy: 0.9838
Epoch 12/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0211 - accuracy:
0.9927 - val_loss: 0.0758 - val_accuracy: 0.9825
Epoch 13/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0185 - accuracy:
0.9938 - val_loss: 0.0768 - val_accuracy: 0.9817
Epoch 14/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0193 - accuracy:
0.9932 - val_loss: 0.0818 - val_accuracy: 0.9821
Epoch 15/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0203 - accuracy:
0.9929 - val_loss: 0.0814 - val_accuracy: 0.9818
Epoch 16/20
469/469 [==============================] - 10s 22ms/step - loss: 0.0173 - accuracy:
0.9943 - val_loss: 0.0842 - val_accuracy: 0.9812
Epoch 17/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0146 - accuracy:
0.9949 - val_loss: 0.0880 - val_accuracy: 0.9820
Epoch 18/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0182 - accuracy:
0.9943 - val_loss: 0.0792 - val_accuracy: 0.9829
Epoch 19/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0167 - accuracy:
0.9943 - val_loss: 0.0836 - val_accuracy: 0.9819
```

```
Epoch 20/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0166 - accuracy: 0.9946 - val_loss: 0.0770 - val_accuracy: 0.9836
```



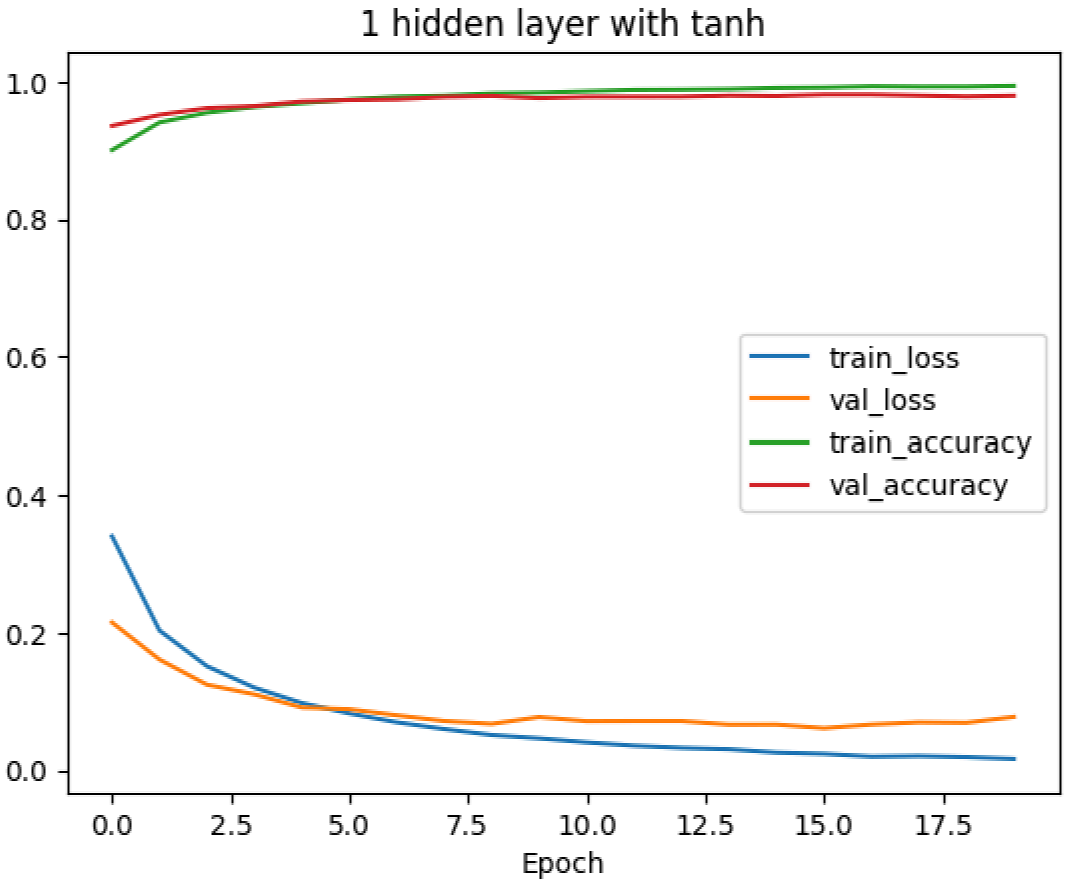
```
1/1 [==============================] - 0s 103ms/step
Model prediction: 7
```

CODE:

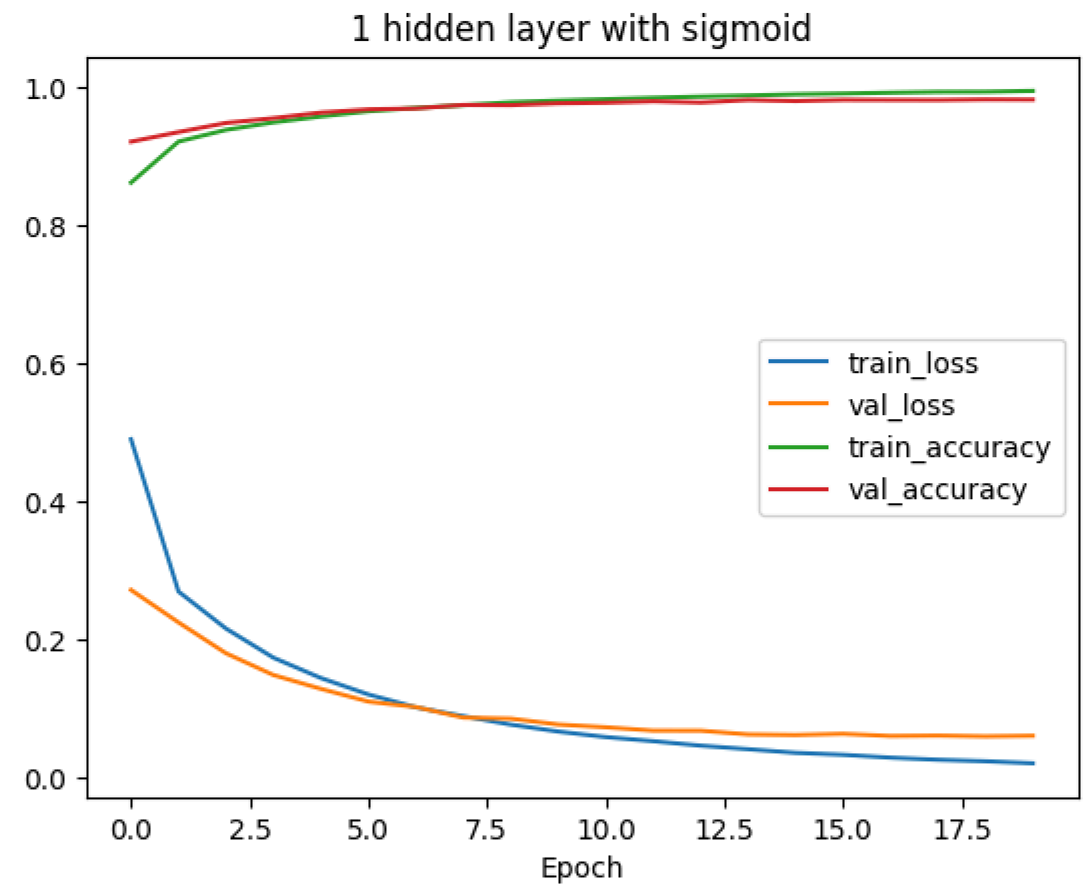
```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

# load MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# normalize pixel values to range [0, 1]
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255
```

```
# convert class labels to binary class matrices
num_classes = 10
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

# create a list of models to train
models = []

# model with 1 hidden layer and tanh activation
model = Sequential()
model.add(Dense(512, activation='tanh', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('1 hidden layer with tanh', model))

# model with 1 hidden layer and sigmoid activation
model = Sequential()
model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('1 hidden layer with sigmoid', model))

# model with 2 hidden layers and tanh activation
model = Sequential()
model.add(Dense(512, activation='tanh', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('2 hidden layers with tanh', model))

# model with 2 hidden layers and sigmoid activation
model = Sequential()
model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='sigmoid'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('2 hidden layers with sigmoid', model))

# train each model and plot loss and accuracy curves
for name, model in models:
    model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
    history = model.fit(x_train.reshape(-1, 784), y_train,
validation_data=(x_test.reshape(-1, 784), y_test),
                        epochs=20, batch_size=128, verbose=0)
    # plot loss and accuracy curves
    plt.plot(history.history['loss'], label='train_loss')
    plt.plot(history.history['val_loss'], label='val_loss')
    plt.plot(history.history['accuracy'], label='train_accuracy')
    plt.plot(history.history['val_accuracy'], label='val_accuracy')
    plt.title(name)
    plt.xlabel('Epoch')
    plt.legend()
    plt.show()

    # evaluate the model on test data
    loss, accuracy = model.evaluate(x_test.reshape(-1, 784), y_test,
verbose=0)
    print('{} - Test loss: {:.4f}, Test accuracy: {:.4f}'.format(name, loss,
accuracy))
```

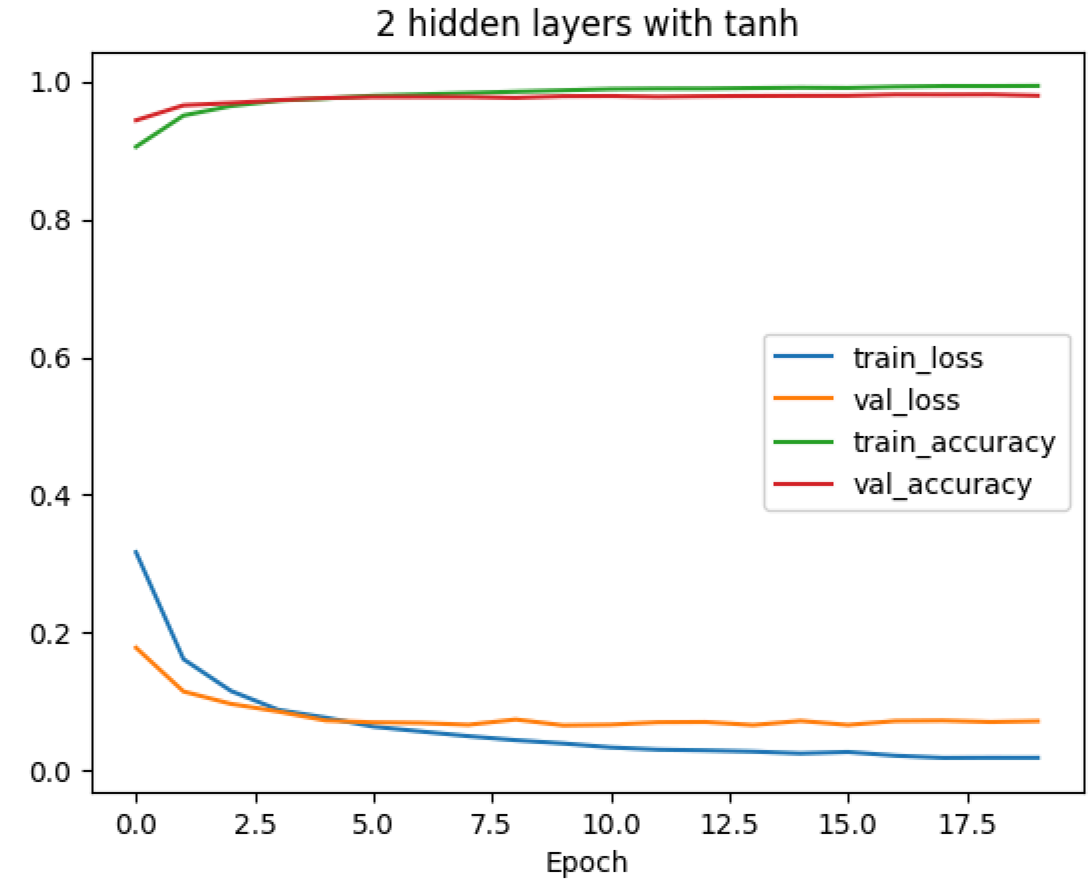
OUTPUT:



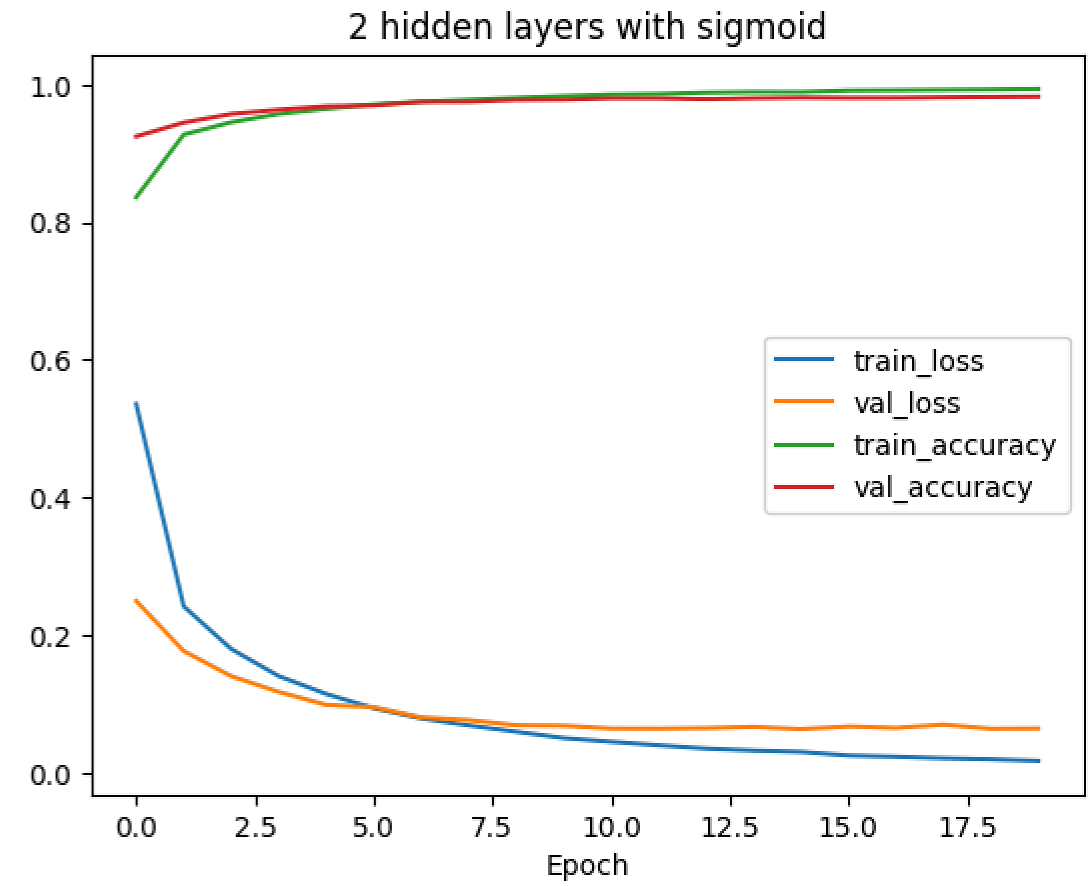
1 hidden layer with tanh - Test loss: 0.0777, Test accuracy: 0.9801



1 hidden layer with sigmoid - Test loss: 0.0615, Test accuracy: 0.9814



2 hidden layers with tanh - Test loss: 0.0718, Test accuracy: 0.9795



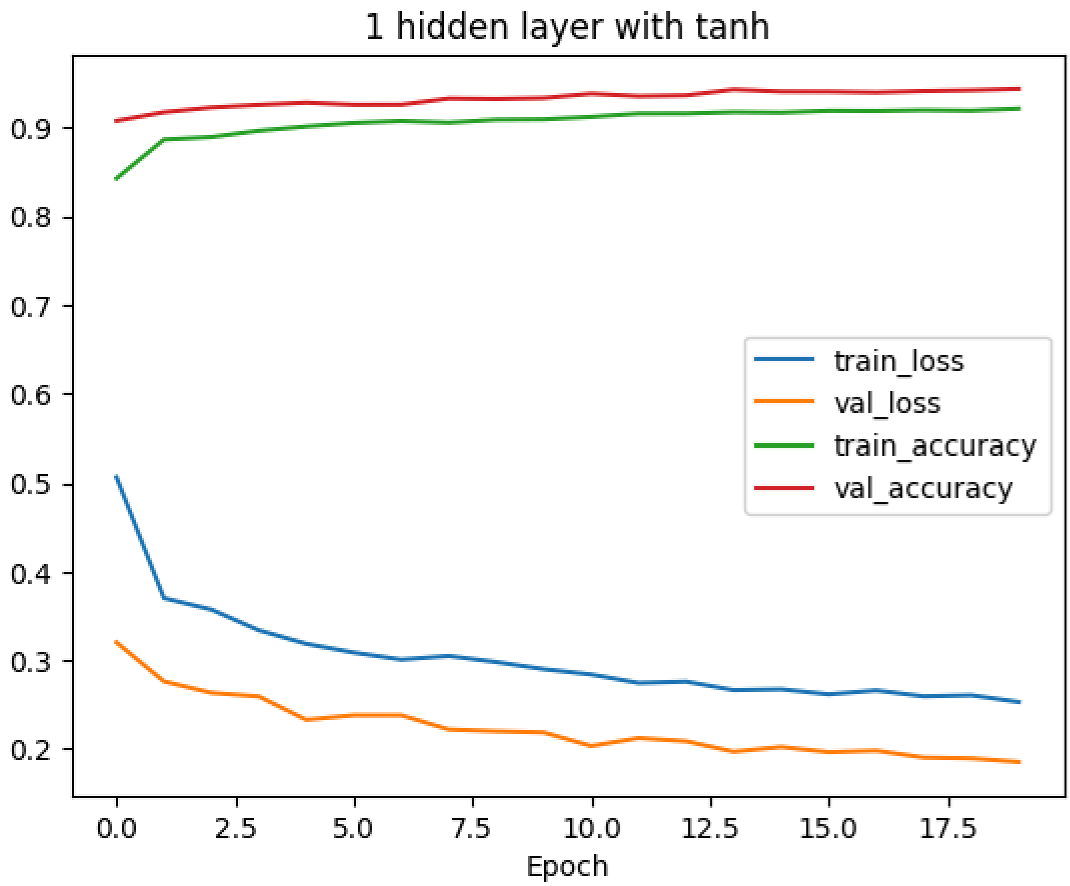
2 hidden layers with sigmoid - Test loss: 0.0645, Test accuracy: 0.9828

CODE:

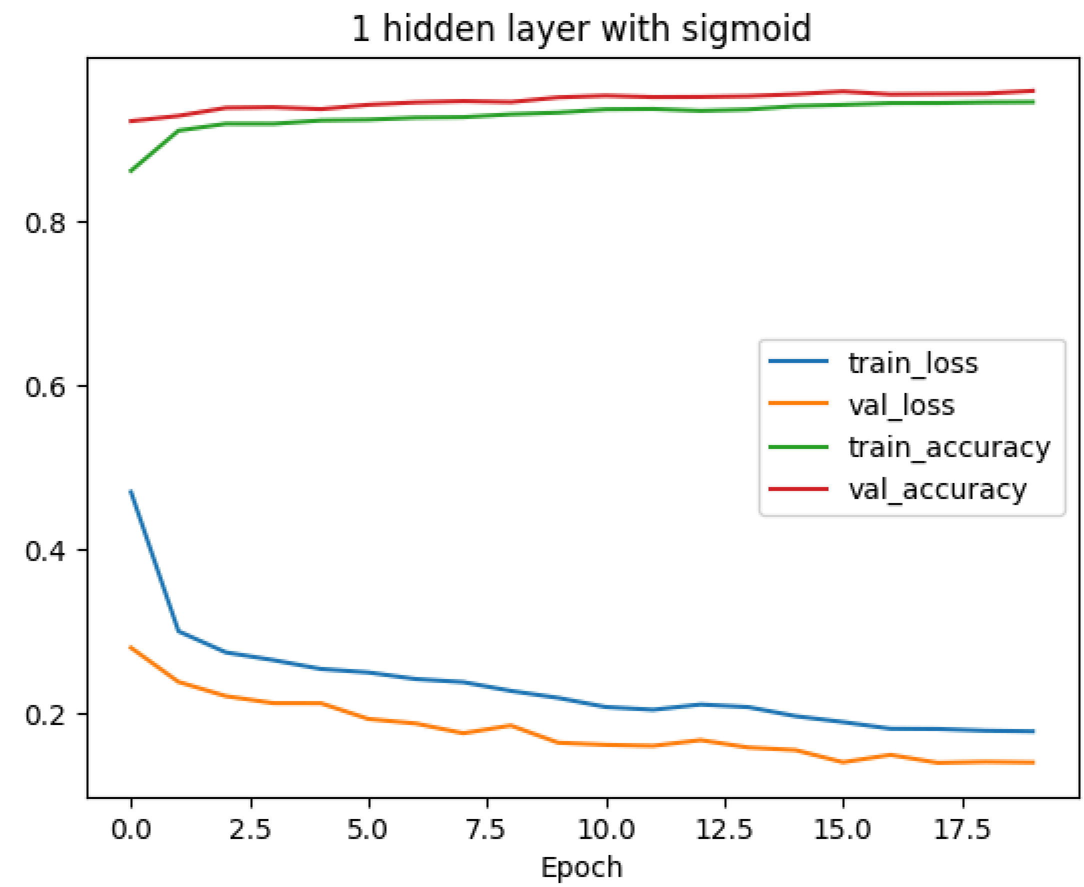
```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

# load MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# convert class labels to binary class matrices
num_classes = 10
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

# create a list of models to train
models = []
```

```
# model with 1 hidden layer and tanh activation
model = Sequential()
model.add(Dense(512, activation='tanh', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('1 hidden layer with tanh', model))

# model with 1 hidden layer and sigmoid activation
model = Sequential()
model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('1 hidden layer with sigmoid', model))

# model with 2 hidden layers and tanh activation
model = Sequential()
model.add(Dense(512, activation='tanh', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('2 hidden layers with tanh', model))

# model with 2 hidden layers and sigmoid activation
model = Sequential()
model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='sigmoid'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('2 hidden layers with sigmoid', model))

# train each model and plot loss and accuracy curves
for name, model in models:
    model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
    history = model.fit(x_train.reshape(-1, 784), y_train,
validation_data=(x_test.reshape(-1, 784), y_test),
                        epochs=20, batch_size=128, verbose=0)
    # plot loss and accuracy curves
    plt.plot(history.history['loss'], label='train_loss')
    plt.plot(history.history['val_loss'], label='val_loss')
    plt.plot(history.history['accuracy'], label='train_accuracy')
    plt.plot(history.history['val_accuracy'], label='val_accuracy')
    plt.title(name)
    plt.xlabel('Epoch')
    plt.legend()
    plt.show()

    # evaluate the model on test data
    loss, accuracy = model.evaluate(x_test.reshape(-1, 784), y_test,
verbose=0)
    print('{} - Test loss: {:.4f}, Test accuracy: {:.4f}'.format(name, loss,
accuracy))
```

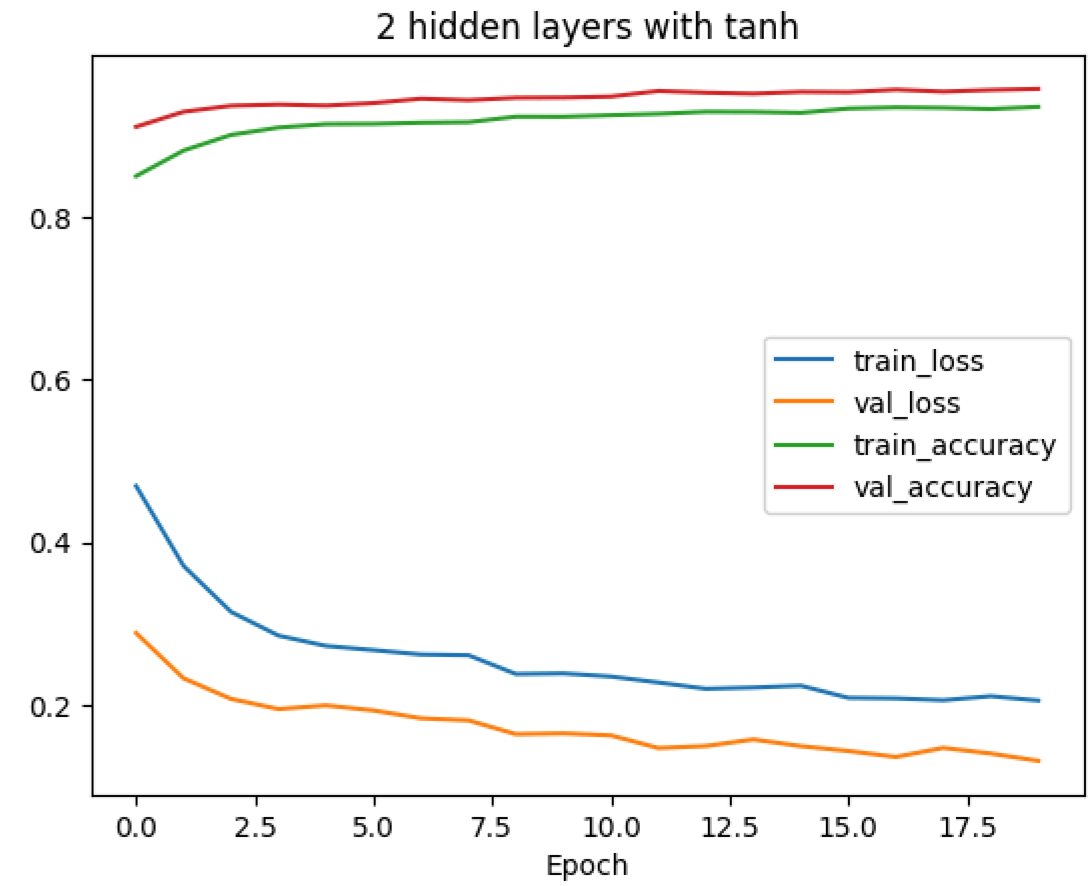
OUTPUT:



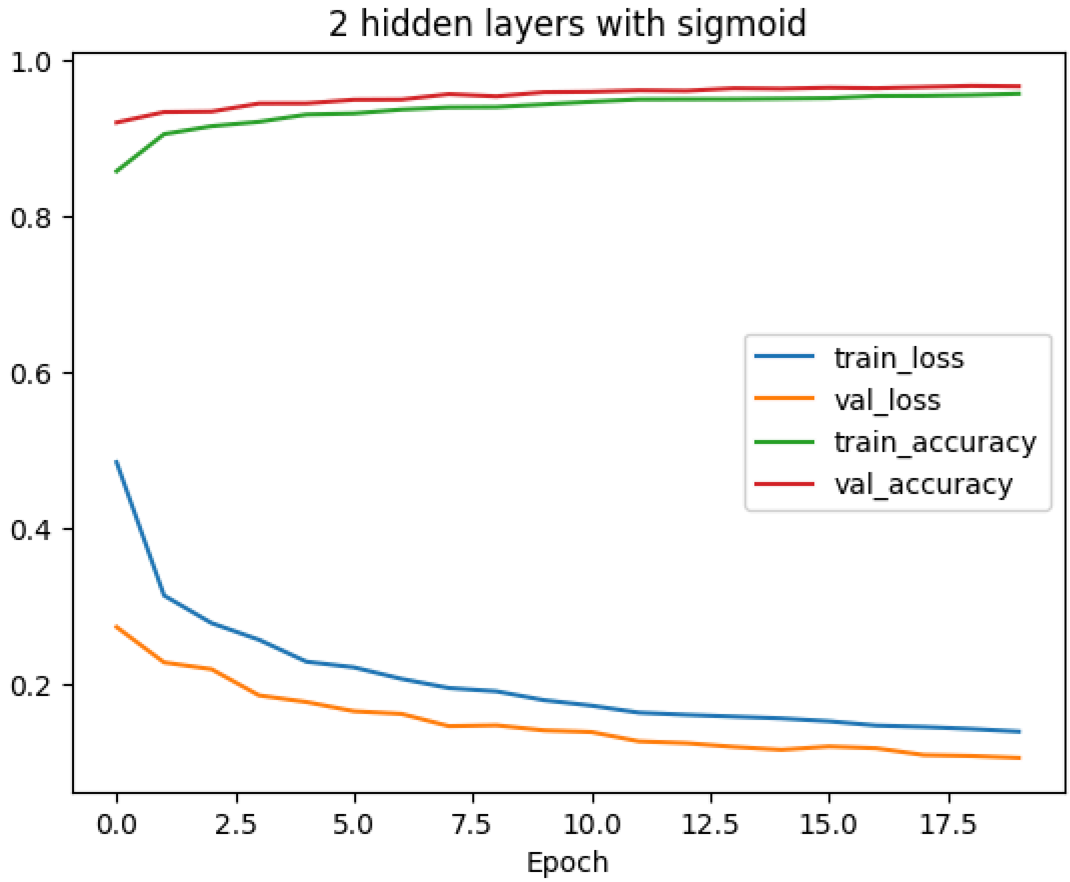
1 hidden layer with tanh - Test loss: 0.1855, Test accuracy: 0.9436



1 hidden layer with sigmoid - Test loss: 0.1407, Test accuracy: 0.9590



2 hidden layers with tanh - Test loss: 0.1309, Test accuracy: 0.9578



2 hidden layers with sigmoid - Test loss: 0.1052, Test accuracy: 0.9660