Use Case Description:

Predicting the diabetes disease Programming elements:

Keras Basics In 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.

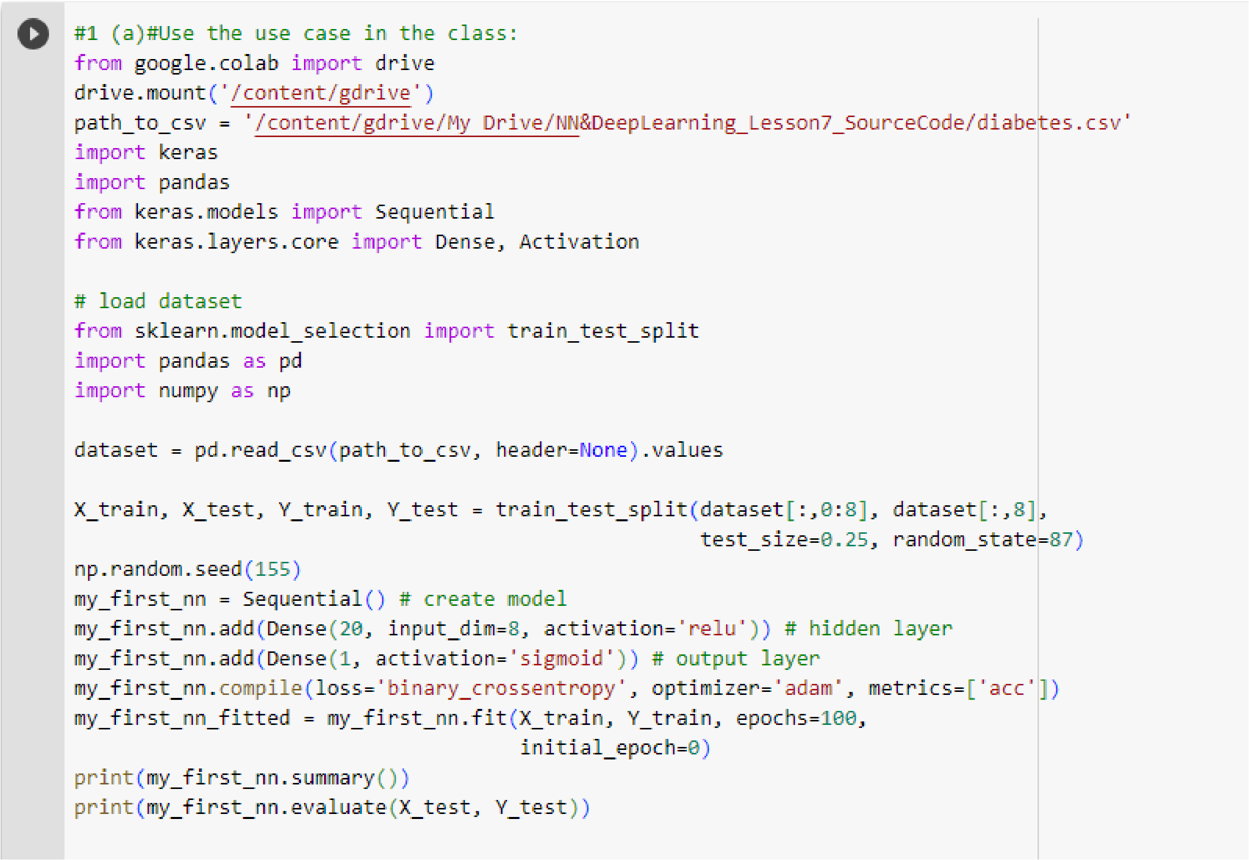
2. Change the data source to Breast Cancer dataset * available in the source code folder and make required changes.

Report accuracy of the model.

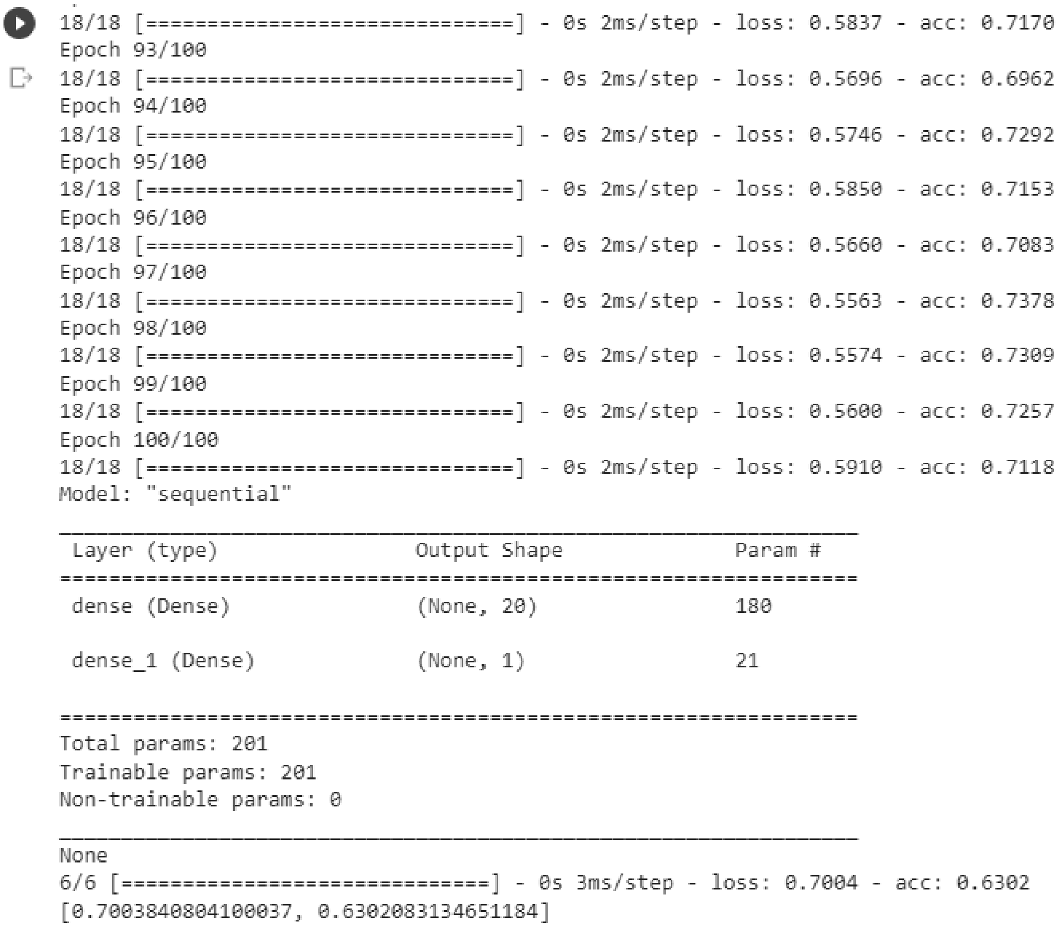
3. Normalize the data before feeding the data to the model and check how the normalization change your accuracy (code given below).

from sklearn.preprocessing import StandardScaler sc = StandardScaler()

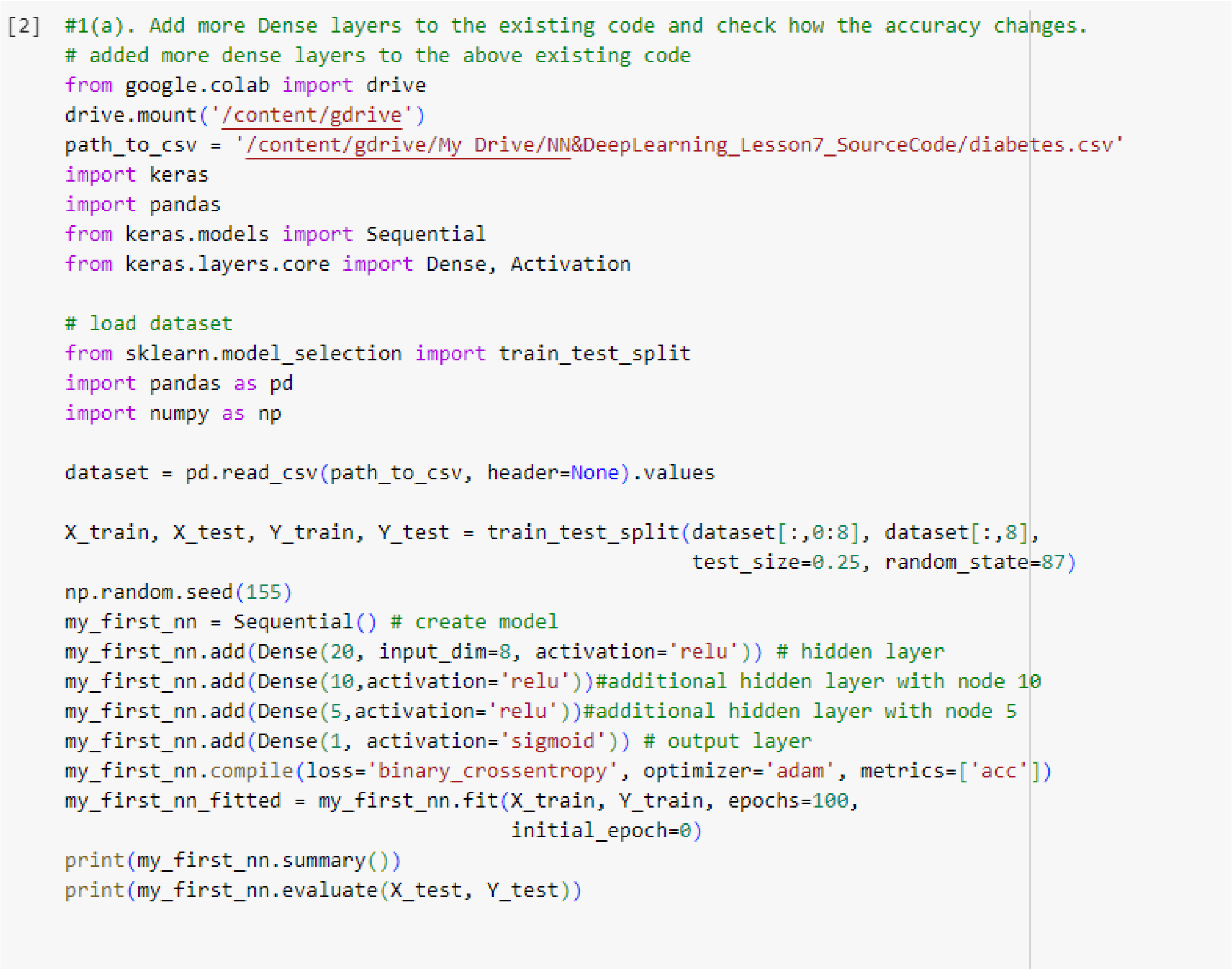
Breast Cancer dataset is designated to predict if a patient has Malignant (M) or Benign = B cancer



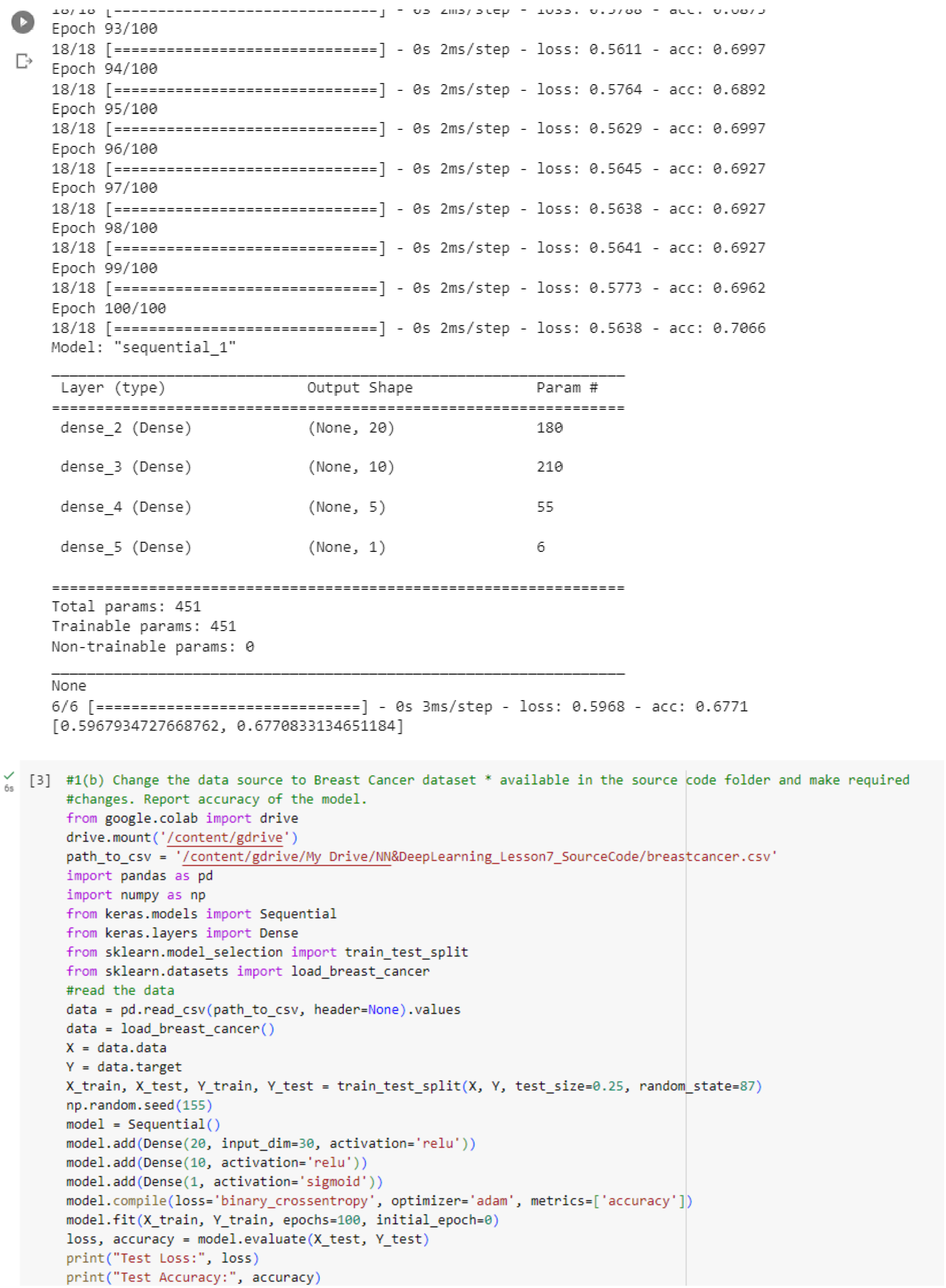
```
#1 (a)#Use the use case in the class:
from google.colab import drive
drive.mount('/content/gdrive')
path_to_csv = '/content/gdrive/My Drive/NN&DeepLearning_Lesson7_SourceCode/diabetes.csv'
import keras
import pandas
from keras.models import Sequential
from keras.layers.core import Dense, Activation

# 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(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))
```



```
18/18 [==============================] - 0s 2ms/step - loss: 0.5837 - acc: 0.7170
Epoch 93/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5696 - acc: 0.6962
Epoch 94/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5746 - acc: 0.7292
Epoch 95/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5850 - acc: 0.7153
Epoch 96/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5660 - acc: 0.7083
Epoch 97/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5563 - acc: 0.7378
Epoch 98/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5574 - acc: 0.7309
Epoch 99/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5600 - acc: 0.7257
Epoch 100/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5910 - acc: 0.7118
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense (Dense)               (None, 20)                180       
                                                                 
 dense_1 (Dense)             (None, 1)                 21        
                                                                 
=================================================================
Total params: 201
Trainable params: 201
Non-trainable params: 0
_________________________________________________________________
None
6/6 [==============================] - 0s 3ms/step - loss: 0.7004 - acc: 0.6302
[0.7003840804100037, 0.6302083134651184]
```

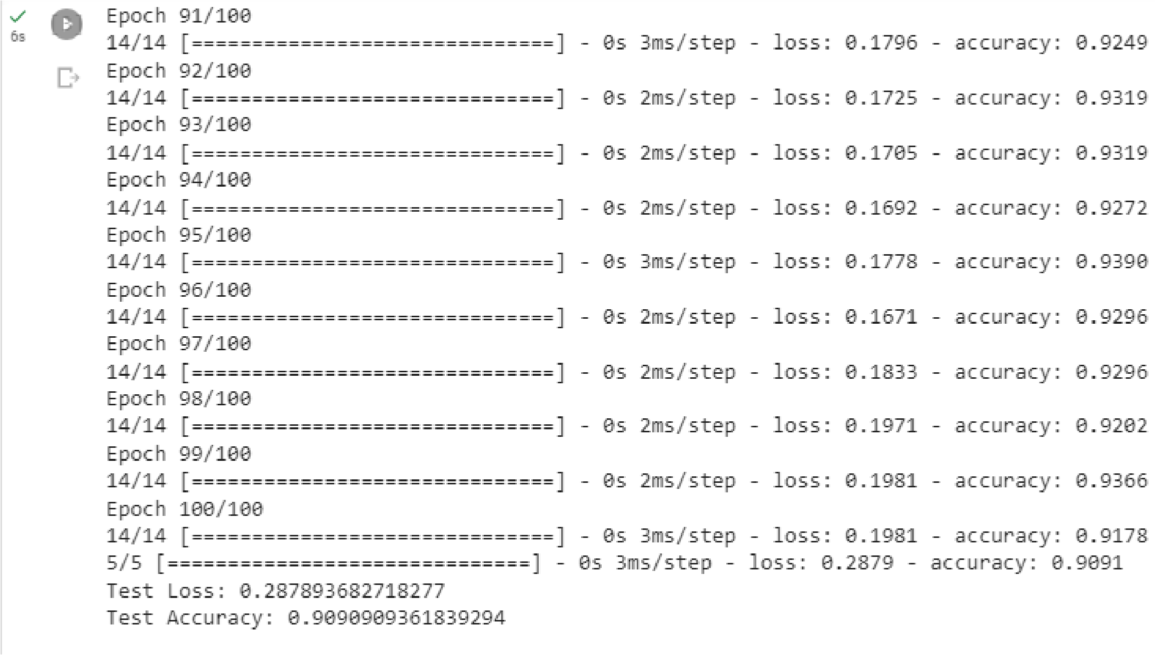


```
#1(a). Add more Dense layers to the existing code and check how the accuracy changes.
# added more dense layers to the above existing code
from google.colab import drive
drive.mount('/content/gdrive')
path_to_csv = '/content/gdrive/My Drive/NN&DeepLearning_Lesson7_SourceCode/diabetes.csv'
import keras
import pandas
from keras.models import Sequential
from keras.layers.core 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(10,activation='relu'))#additional hidden layer with node 10
my_first_nn.add(Dense(5,activation='relu'))#additional hidden layer with node 5
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))
```

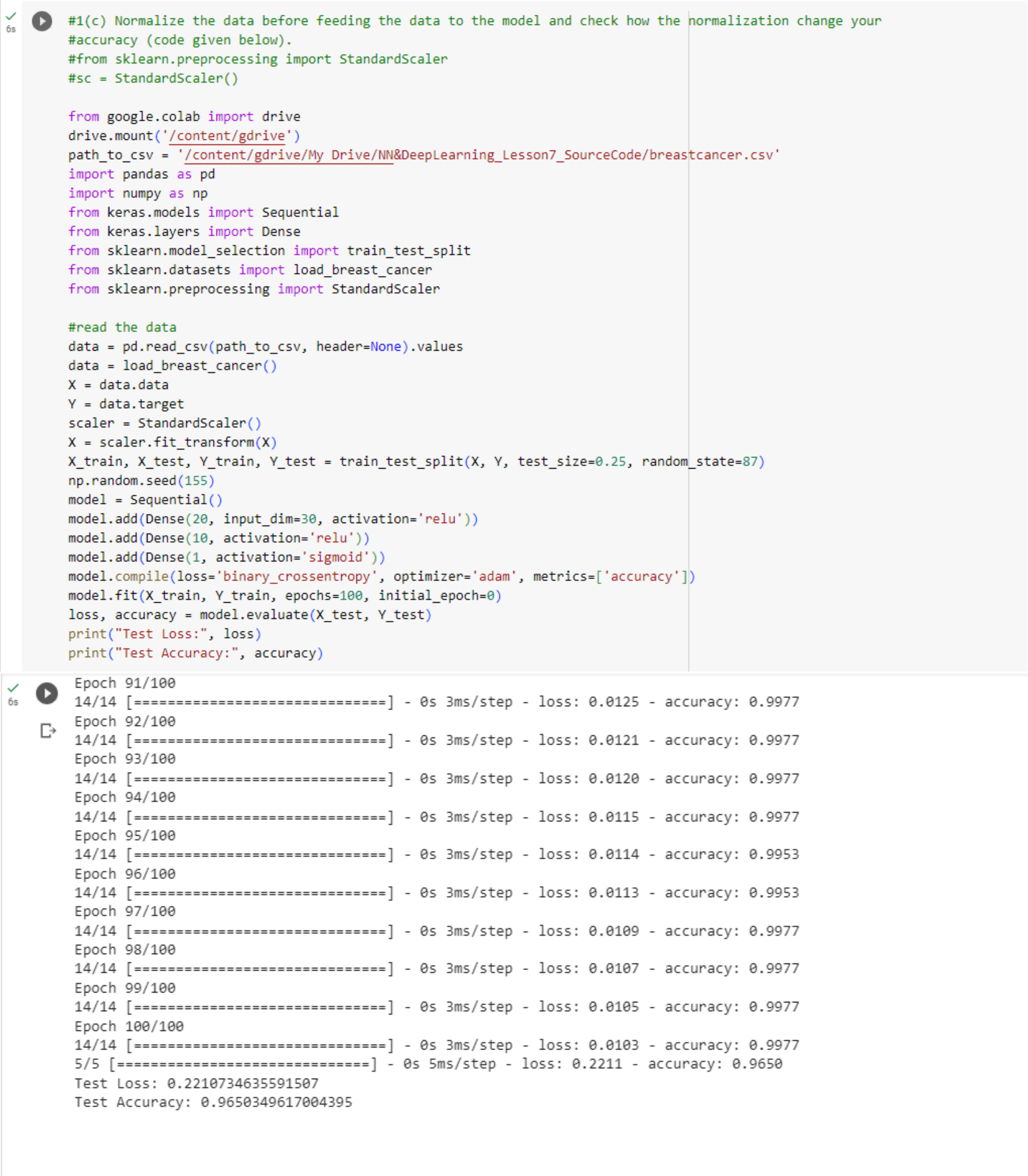


```
18/18 [==============================] - 0s 2ms/step - loss: 0.5788 - acc: 0.6875
Epoch 93/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5611 - acc: 0.6997
Epoch 94/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5764 - acc: 0.6892
Epoch 95/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5629 - acc: 0.6997
Epoch 96/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5645 - acc: 0.6927
Epoch 97/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5638 - acc: 0.6927
Epoch 98/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5641 - acc: 0.6927
Epoch 99/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5773 - acc: 0.6962
Epoch 100/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5638 - acc: 0.7066
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense_2 (Dense)             (None, 20)                180       
                                                                 
 dense_3 (Dense)             (None, 10)                210       
                                                                 
 dense_4 (Dense)             (None, 5)                 55        
                                                                 
 dense_5 (Dense)             (None, 1)                 6         
                                                                 
=================================================================
Total params: 451
Trainable params: 451
Non-trainable params: 0
_________________________________________________________________
None
6/6 [==============================] - 0s 3ms/step - loss: 0.5968 - acc: 0.6771
[0.5967934727668762, 0.6770833134651184]
```

```
[3] #1(b) Change the data source to Breast Cancer dataset * available in the source code folder and make required
    #changes. Report accuracy of the model.
    from google.colab import drive
    drive.mount('/content/gdrive')
    path_to_csv = '/content/gdrive/My Drive/NN&DeepLearning_Lesson7_SourceCode/breastcancer.csv'
    import pandas as pd
    import numpy as np
    from keras.models import Sequential
    from keras.layers import Dense
    from sklearn.model_selection import train_test_split
    from sklearn.datasets import load_breast_cancer
    #read the data
    data = pd.read_csv(path_to_csv, header=None).values
    data = load_breast_cancer()
    X = data.data
    Y = data.target
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=87)
    np.random.seed(155)
    model = Sequential()
    model.add(Dense(20, input_dim=30, activation='relu'))
    model.add(Dense(10, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    model.fit(X_train, Y_train, epochs=100, initial_epoch=0)
    loss, accuracy = model.evaluate(X_test, Y_test)
    print("Test Loss:", loss)
    print("Test Accuracy:", accuracy)
```



```
Epoch 91/100
14/14 [==============================] - 0s 3ms/step - loss: 0.1796 - accuracy: 0.9249
Epoch 92/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1725 - accuracy: 0.9319
Epoch 93/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1705 - accuracy: 0.9319
Epoch 94/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1692 - accuracy: 0.9272
Epoch 95/100
14/14 [==============================] - 0s 3ms/step - loss: 0.1778 - accuracy: 0.9390
Epoch 96/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1671 - accuracy: 0.9296
Epoch 97/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1833 - accuracy: 0.9296
Epoch 98/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1971 - accuracy: 0.9202
Epoch 99/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1981 - accuracy: 0.9366
Epoch 100/100
14/14 [==============================] - 0s 3ms/step - loss: 0.1981 - accuracy: 0.9178
5/5 [==============================] - 0s 3ms/step - loss: 0.2879 - accuracy: 0.9091
Test Loss: 0.287893682718277
Test Accuracy: 0.9090909361839294
```



```
#1(c) 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()

from google.colab import drive
drive.mount('/content/gdrive')
path_to_csv = '/content/gdrive/My Drive/NN&DeepLearning_Lesson7_SourceCode/breastcancer.csv'
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_breast_cancer
from sklearn.preprocessing import StandardScaler

#read the data
data = pd.read_csv(path_to_csv, header=None).values
data = load_breast_cancer()
X = data.data
Y = data.target
scaler = StandardScaler()
X = scaler.fit_transform(X)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=87)
np.random.seed(155)
model = Sequential()
model.add(Dense(20, input_dim=30, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X_train, Y_train, epochs=100, initial_epoch=0)
loss, accuracy = model.evaluate(X_test, Y_test)
print("Test Loss:", loss)
print("Test Accuracy:", accuracy)
```

```
Epoch 91/100
14/14 [==============================] - 0s 3ms/step - loss: 0.0125 - accuracy: 0.9977
Epoch 92/100
14/14 [==============================] - 0s 3ms/step - loss: 0.0121 - accuracy: 0.9977
Epoch 93/100
14/14 [==============================] - 0s 3ms/step - loss: 0.0120 - accuracy: 0.9977
Epoch 94/100
14/14 [==============================] - 0s 3ms/step - loss: 0.0115 - accuracy: 0.9977
Epoch 95/100
14/14 [==============================] - 0s 3ms/step - loss: 0.0114 - accuracy: 0.9953
Epoch 96/100
14/14 [==============================] - 0s 3ms/step - loss: 0.0113 - accuracy: 0.9953
Epoch 97/100
14/14 [==============================] - 0s 3ms/step - loss: 0.0109 - accuracy: 0.9977
Epoch 98/100
14/14 [==============================] - 0s 3ms/step - loss: 0.0107 - accuracy: 0.9977
Epoch 99/100
14/14 [==============================] - 0s 3ms/step - loss: 0.0105 - accuracy: 0.9977
Epoch 100/100
14/14 [==============================] - 0s 3ms/step - loss: 0.0103 - accuracy: 0.9977
5/5 [==============================] - 0s 5ms/step - loss: 0.2211 - accuracy: 0.9650
Test Loss: 0.2210734635591507
Test Accuracy: 0.9650349617004395
```