Neural Network Deep Learning

Assignment – 6

Name: Kishor Kumar Andekar

Student ID: 700744713

Github Link : https://github.com/kishorreyansh/Neural-Network-Deep-Learning/tree/main/Assignment-6

Use Case Description: Predicting the diabetes disease.

Programming elements: Keras Basics

1.Use the use case in the class:

```
import pandas as pd
import numpy as npP
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split

# Load the diabetes dataset
diabetes_data = pd.read_csv("diabetes.csv", header=None).values

# Split the dataset into features and target variables
features = diabetes_data[:, 0:8]
target = diabetes_data[:, 8]

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.25, random_state=87)

# Set the random seed for reproducibility
np.random.seed(155)

# Define the architecture of the neural network model
model = Sequential()  # Initialize the model
model.add(Dense(20, input_dim=8, activation='relu'))  # Add a hidden layer
model.add(Dense(1, activation='sigmoid'))  # Add an output layer
```

```python
# Compile the model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model
history = model.fit(X_train, y_train, epochs=100, initial_epoch=0, verbose=1)

# Print model summary
print("Model Summary:")
print(model.summary())

# Evaluate the model on the test set
evaluation_result = model.evaluate(X_test, y_test)
print("\nEvaluation Result (loss, accuracy):", evaluation_result)
```

Output:

```
Epoch 1/100
18/18 [==============================] - 1s 2ms/step - loss: 5.6585 - accuracy: 0.6493
Epoch 2/100
18/18 [==============================] - 0s 2ms/step - loss: 2.9222 - accuracy: 0.5122
Epoch 3/100
18/18 [==============================] - 0s 2ms/step - loss: 2.4026 - accuracy: 0.5955
Epoch 4/100
18/18 [==============================] - 0s 2ms/step - loss: 2.2111 - accuracy: 0.6007
Epoch 5/100
18/18 [==============================] - 0s 2ms/step - loss: 2.0478 - accuracy: 0.6042
Epoch 6/100
18/18 [==============================] - 0s 2ms/step - loss: 1.8848 - accuracy: 0.6233
Epoch 7/100
18/18 [==============================] - 0s 2ms/step - loss: 1.7489 - accuracy: 0.6372
Epoch 8/100
18/18 [==============================] - 0s 2ms/step - loss: 1.6228 - accuracy: 0.6337
Epoch 9/100
18/18 [==============================] - 0s 2ms/step - loss: 1.5255 - accuracy: 0.6389
Epoch 10/100
18/18 [==============================] - 0s 2ms/step - loss: 1.4097 - accuracy: 0.6441
Epoch 11/100
18/18 [==============================] - 0s 2ms/step - loss: 1.3101 - accuracy: 0.6476
Epoch 12/100
18/18 [==============================] - 0s 2ms/step - loss: 1.1902 - accuracy: 0.6493
Epoch 13/100
18/18 [==============================] - 0s 2ms/step - loss: 1.1289 - accuracy: 0.6510
Epoch 14/100
18/18 [==============================] - 0s 2ms/step - loss: 1.0788 - accuracy: 0.6545
Epoch 15/100
18/18 [==============================] - 0s 2ms/step - loss: 0.9764 - accuracy: 0.6684
```

Connected to Python 3 Google Compute Engine backend

```
Epoch 16/100
18/18 [==============================] - 0s 2ms/step - loss: 0.9292 - accuracy: 0.6667
Epoch 17/100
18/18 [==============================] - 0s 2ms/step - loss: 0.8578 - accuracy: 0.6719
Epoch 18/100
18/18 [==============================] - 0s 2ms/step - loss: 0.8143 - accuracy: 0.6632
Epoch 19/100
18/18 [==============================] - 0s 2ms/step - loss: 0.7551 - accuracy: 0.6701
Epoch 20/100
18/18 [==============================] - 0s 2ms/step - loss: 0.7270 - accuracy: 0.6753
Epoch 21/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6878 - accuracy: 0.6736
Epoch 22/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6593 - accuracy: 0.6823
Epoch 23/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6637 - accuracy: 0.6788
Epoch 24/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6320 - accuracy: 0.6771
Epoch 25/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6220 - accuracy: 0.6910
Epoch 26/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6128 - accuracy: 0.6788
Epoch 27/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6059 - accuracy: 0.6823
Epoch 28/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6018 - accuracy: 0.6910
Epoch 29/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5985 - accuracy: 0.6910
Epoch 30/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5945 - accuracy: 0.6910
```

```
Epoch 31/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5888 - accuracy: 0.6910
Epoch 32/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5861 - accuracy: 0.6927
Epoch 33/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5960 - accuracy: 0.6979
Epoch 34/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5913 - accuracy: 0.6997
Epoch 35/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6474 - accuracy: 0.6545
Epoch 36/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6593 - accuracy: 0.6823
Epoch 37/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6030 - accuracy: 0.6875
Epoch 38/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5782 - accuracy: 0.7049
Epoch 39/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5871 - accuracy: 0.7257
Epoch 40/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5945 - accuracy: 0.7101
Epoch 41/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5752 - accuracy: 0.7066
Epoch 42/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5726 - accuracy: 0.6979
Epoch 43/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5738 - accuracy: 0.7101
Epoch 44/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5861 - accuracy: 0.7066
Epoch 45/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5763 - accuracy: 0.7153
```

```
Epoch 46/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5817 - accuracy: 0.7014
Epoch 47/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5754 - accuracy: 0.7031
Epoch 48/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5790 - accuracy: 0.6927
Epoch 49/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5920 - accuracy: 0.7014
Epoch 50/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5997 - accuracy: 0.6979
Epoch 51/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5973 - accuracy: 0.6944
Epoch 52/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6000 - accuracy: 0.7083
Epoch 53/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5656 - accuracy: 0.6997
Epoch 54/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5651 - accuracy: 0.7049
Epoch 55/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5809 - accuracy: 0.7066
Epoch 56/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6000 - accuracy: 0.7066
Epoch 57/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5787 - accuracy: 0.7066
Epoch 58/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5634 - accuracy: 0.7135
Epoch 59/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5675 - accuracy: 0.7292
Epoch 60/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5648 - accuracy: 0.7222
```

```
Epoch 61/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5651 - accuracy: 0.7101
Epoch 62/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5591 - accuracy: 0.7396
Epoch 63/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5893 - accuracy: 0.7083
Epoch 64/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5660 - accuracy: 0.7188
Epoch 65/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5522 - accuracy: 0.7344
Epoch 66/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5503 - accuracy: 0.7378
Epoch 67/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6131 - accuracy: 0.6979
Epoch 68/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5597 - accuracy: 0.7257
Epoch 69/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5711 - accuracy: 0.7188
Epoch 70/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5782 - accuracy: 0.7257
Epoch 71/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5821 - accuracy: 0.7205
Epoch 72/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5615 - accuracy: 0.7135
Epoch 73/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5703 - accuracy: 0.7344
Epoch 74/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5531 - accuracy: 0.7274
Epoch 75/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5680 - accuracy: 0.7170
```

```
Epoch 76/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5711 - accuracy: 0.7170
Epoch 77/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5837 - accuracy: 0.7188
Epoch 78/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5681 - accuracy: 0.7170
Epoch 79/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5564 - accuracy: 0.7118
Epoch 80/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5923 - accuracy: 0.6997
Epoch 81/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6096 - accuracy: 0.6806
Epoch 82/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5718 - accuracy: 0.7222
Epoch 83/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5484 - accuracy: 0.7205
Epoch 84/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5687 - accuracy: 0.7031
Epoch 85/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5577 - accuracy: 0.7170
Epoch 86/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5616 - accuracy: 0.7188
Epoch 87/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5804 - accuracy: 0.7292
Epoch 88/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5911 - accuracy: 0.6944
Epoch 89/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5565 - accuracy: 0.7240
Epoch 90/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5605 - accuracy: 0.7257
```

```
18/18 [==============================] - 0s 2ms/step - loss: 0.5390 - accuracy: 0.7309
Epoch 96/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5483 - accuracy: 0.7222
Epoch 97/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5441 - accuracy: 0.7240
Epoch 98/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5522 - accuracy: 0.7188
Epoch 99/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5567 - accuracy: 0.7153
Epoch 100/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5514 - accuracy: 0.7205
Model Summary:
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_2 (Dense)             (None, 20)                180

 dense_3 (Dense)             (None, 1)                 21

=================================================================
Total params: 201 (804.00 Byte)
Trainable params: 201 (804.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
6/6 [==============================] - 0s 3ms/step - loss: 0.6131 - accuracy: 0.6562

Evaluation Result (loss, accuracy): [0.6131137013435364, 0.65625]
```

1. a. Change the data source to Breast Cancer dataset * available in the source code folder and make required changes. Report accuracy of the model

```
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

# Load dataset
diabetes_data = pd.read_csv("diabetes.csv", header=None).values

# Split dataset into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(diabetes_data[:,0:8], diabetes_data[:,8],
                                                    test_size=0.25, random_state=87)

# Set random seed for reproducibility
np.random.seed(155)

# Define the architecture of the neural network model
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
my_first_nn.add(Dense(5, activation='relu'))   # Additional hidden layer
my_first_nn.add(Dense(1, activation='sigmoid'))  # output layer

# Compile the model
my_first_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
```

```
# Compile the model
my_first_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])

# Train the model
my_first_nn_fitted = my_first_nn.fit(X_train, Y_train, epochs=100, initial_epoch=0)

# Print model summary
print("Model Summary:")
print(my_first_nn.summary())

# Evaluate the model on the test set
evaluation_result = my_first_nn.evaluate(X_test, Y_test)
print("\nEvaluation Result (loss, accuracy):", evaluation_result)
```

OUTPUT:

```
Epoch 1/100
18/18 [==============================] - 1s 3ms/step - loss: 6.3937 - acc: 0.3299
Epoch 2/100
18/18 [==============================] - 0s 2ms/step - loss: 2.5636 - acc: 0.3628
Epoch 3/100
18/18 [==============================] - 0s 2ms/step - loss: 1.2427 - acc: 0.3646
Epoch 4/100
18/18 [==============================] - 0s 2ms/step - loss: 0.9662 - acc: 0.4705
Epoch 5/100
18/18 [==============================] - 0s 2ms/step - loss: 0.8481 - acc: 0.5451
Epoch 6/100
18/18 [==============================] - 0s 2ms/step - loss: 0.7971 - acc: 0.5729
Epoch 7/100
18/18 [==============================] - 0s 2ms/step - loss: 0.7572 - acc: 0.6024
Epoch 8/100
18/18 [==============================] - 0s 2ms/step - loss: 0.7296 - acc: 0.6215
Epoch 9/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6971 - acc: 0.6337
Epoch 10/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6884 - acc: 0.6736
Epoch 11/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6723 - acc: 0.6719
Epoch 12/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6596 - acc: 0.6736
Epoch 13/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6570 - acc: 0.6719
Epoch 14/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6525 - acc: 0.6753
Epoch 18/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6414 - acc: 0.6806
Epoch 19/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6400 - acc: 0.6788
Epoch 20/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6385 - acc: 0.6840
Epoch 21/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6337 - acc: 0.6892
Epoch 22/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6337 - acc: 0.6892
Epoch 23/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6344 - acc: 0.6806
Epoch 24/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6301 - acc: 0.6944
Epoch 25/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6284 - acc: 0.6892
Epoch 26/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6260 - acc: 0.6892
Epoch 27/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6257 - acc: 0.6892
Epoch 28/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6336 - acc: 0.6788
Epoch 29/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6312 - acc: 0.6840
Epoch 30/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6362 - acc: 0.6823
Epoch 31/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6252 - acc: 0.6771
Epoch 32/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6228 - acc: 0.6823
```

```
Epoch 33/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6175 - acc: 0.6840
Epoch 34/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6276 - acc: 0.6823
Epoch 35/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6154 - acc: 0.6927
Epoch 36/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6206 - acc: 0.6892
Epoch 37/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6150 - acc: 0.6979
Epoch 38/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6254 - acc: 0.6858
Epoch 39/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6182 - acc: 0.6927
Epoch 40/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6167 - acc: 0.6840
Epoch 41/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6337 - acc: 0.6788
Epoch 42/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6292 - acc: 0.6719
Epoch 43/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6279 - acc: 0.6684
Epoch 44/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6273 - acc: 0.6667
Epoch 45/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6271 - acc: 0.6667
Epoch 46/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6272 - acc: 0.6684
Epoch 47/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6270 - acc: 0.6684
```

```
Epoch 48/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6266 - acc: 0.6684
Epoch 49/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6272 - acc: 0.6719
Epoch 50/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6260 - acc: 0.6684
Epoch 51/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6257 - acc: 0.6684
Epoch 52/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6262 - acc: 0.6684
Epoch 53/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6253 - acc: 0.6649
Epoch 54/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6258 - acc: 0.6684
Epoch 55/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6219 - acc: 0.6719
Epoch 56/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6180 - acc: 0.6719
Epoch 57/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6123 - acc: 0.6910
Epoch 58/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6097 - acc: 0.6892
Epoch 59/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6097 - acc: 0.6892
Epoch 60/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6089 - acc: 0.6927
Epoch 61/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6103 - acc: 0.6875
Epoch 62/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6159 - acc: 0.6927
```

```
Epoch 63/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6059 - acc: 0.6892
Epoch 64/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6053 - acc: 0.6962
Epoch 65/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6077 - acc: 0.6910
Epoch 66/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6095 - acc: 0.6892
Epoch 67/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6075 - acc: 0.6927
Epoch 68/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6032 - acc: 0.6997
Epoch 69/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6061 - acc: 0.6875
Epoch 70/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6114 - acc: 0.6962
Epoch 71/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6090 - acc: 0.6944
Epoch 72/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6089 - acc: 0.6875
Epoch 73/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6069 - acc: 0.6944
Epoch 74/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6099 - acc: 0.6944
Epoch 75/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6099 - acc: 0.6892
Epoch 76/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6088 - acc: 0.6875
Epoch 77/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6037 - acc: 0.6875
```

```
Epoch 78/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6185 - acc: 0.7014
Epoch 79/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6093 - acc: 0.6910
Epoch 80/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6023 - acc: 0.6875
Epoch 81/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6071 - acc: 0.6944
Epoch 82/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5972 - acc: 0.7031
Epoch 83/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6093 - acc: 0.6979
Epoch 84/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5983 - acc: 0.7014
Epoch 85/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5977 - acc: 0.6944
Epoch 86/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6047 - acc: 0.7014
Epoch 87/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5970 - acc: 0.6927
Epoch 88/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5964 - acc: 0.6979
Epoch 89/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6042 - acc: 0.6962
Epoch 90/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5973 - acc: 0.7031
Epoch 91/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6100 - acc: 0.6892
Epoch 92/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6216 - acc: 0.6858
```

```
18/18 [==============================] - 0s 2ms/step - loss: 0.5859 - acc: 0.7083
Epoch 99/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5939 - acc: 0.7049
Epoch 100/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5991 - acc: 0.7014
Model Summary:
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_4 (Dense)             (None, 20)                180

 dense_5 (Dense)             (None, 10)                210

 dense_6 (Dense)             (None, 5)                 55

 dense_7 (Dense)             (None, 1)                 6

=================================================================
Total params: 451 (1.76 KB)
Trainable params: 451 (1.76 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
6/6 [==============================] - 0s 3ms/step - loss: 0.6767 - acc: 0.6146

Evaluation Result (loss, accuracy): [0.6766515374183655, 0.6145833134651184]
```

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

```
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.preprocessing import LabelEncoder

# Load dataset
dataset = pd.read_csv("Breas Cancer.csv")
dataset.drop(['id'], axis=1, inplace=True)
del dataset['Unnamed: 32']

# Separate features and target variable
X = dataset.iloc[:, 2:].values
Y = dataset.iloc[:, 1].values

# Encode target variable
label = LabelEncoder()
Y = label.fit_transform(Y)

# Split dataset into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25)

# Set random seed for reproducibility
np.random.seed(155)

# Define the architecture of the neural network model
```

```python
# Define the architecture of the neural network model
my_first_nn = Sequential()  # create model
my_first_nn.add(Dense(20, input_dim=29, activation='relu'))  # hidden layer
my_first_nn.add(Dense(1, activation='sigmoid'))  # output layer

# Compile the model
my_first_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])

# Train the model
my_first_nn_fitted = my_first_nn.fit(X_train, Y_train, epochs=100, initial_epoch=0)

# Print model summary
print("Model Summary:")
print(my_first_nn.summary())

# Evaluate the model on the test set
evaluation_result = my_first_nn.evaluate(X_test, Y_test)
print("\nEvaluation Result (loss, accuracy):", evaluation_result)
```

Output:

```
[ ] Epoch 1/100
    14/14 [==============================] - 1s 2ms/step - loss: 33430.8438 - acc: 0.0023
    Epoch 2/100
    14/14 [==============================] - 0s 2ms/step - loss: -2695.1038 - acc: 0.0023
    Epoch 3/100
    14/14 [==============================] - 0s 2ms/step - loss: -37662.5391 - acc: 0.0023
    Epoch 4/100
    14/14 [==============================] - 0s 2ms/step - loss: -73580.8984 - acc: 0.0023
    Epoch 5/100
    14/14 [==============================] - 0s 2ms/step - loss: -110737.2266 - acc: 0.0023
    Epoch 6/100
    14/14 [==============================] - 0s 2ms/step - loss: -146920.6875 - acc: 0.0023
    Epoch 7/100
    14/14 [==============================] - 0s 2ms/step - loss: -185378.5000 - acc: 0.0023
    Epoch 8/100
    14/14 [==============================] - 0s 2ms/step - loss: -225645.4219 - acc: 0.0023
    Epoch 9/100
    14/14 [==============================] - 0s 2ms/step - loss: -268344.3750 - acc: 0.0023
    Epoch 10/100
    14/14 [==============================] - 0s 2ms/step - loss: -313116.0312 - acc: 0.0023
    Epoch 11/100
    14/14 [==============================] - 0s 2ms/step - loss: -360405.2500 - acc: 0.0023
    Epoch 12/100
    14/14 [==============================] - 0s 2ms/step - loss: -410567.0625 - acc: 0.0023
    Epoch 13/100
    14/14 [==============================] - 0s 2ms/step - loss: -463141.9688 - acc: 0.0023
    Epoch 14/100
    14/14 [==============================] - 0s 2ms/step - loss: -520723.7188 - acc: 0.0023
    Epoch 15/100
    14/14 [==============================] - 0s 2ms/step - loss: -579611.5000 - acc: 0.0023
```

```
Epoch 16/100
14/14 [==============================] - 0s 2ms/step - loss: -644077.6875 - acc: 0.0023
Epoch 17/100
14/14 [==============================] - 0s 2ms/step - loss: -708162.7500 - acc: 0.0023
Epoch 18/100
14/14 [==============================] - 0s 2ms/step - loss: -778919.8125 - acc: 0.0023
Epoch 19/100
14/14 [==============================] - 0s 2ms/step - loss: -851612.5000 - acc: 0.0023
Epoch 20/100
14/14 [==============================] - 0s 3ms/step - loss: -926854.5625 - acc: 0.0023
Epoch 21/100
14/14 [==============================] - 0s 2ms/step - loss: -1006678.2500 - acc: 0.0023
Epoch 22/100
14/14 [==============================] - 0s 2ms/step - loss: -1090884.7500 - acc: 0.0023
Epoch 23/100
14/14 [==============================] - 0s 2ms/step - loss: -1175071.7500 - acc: 0.0023
Epoch 24/100
14/14 [==============================] - 0s 2ms/step - loss: -1267992.2500 - acc: 0.0023
Epoch 25/100
14/14 [==============================] - 0s 2ms/step - loss: -1359212.7500 - acc: 0.0023
Epoch 26/100
14/14 [==============================] - 0s 2ms/step - loss: -1455962.0000 - acc: 0.0023
Epoch 27/100
14/14 [==============================] - 0s 2ms/step - loss: -1556434.8750 - acc: 0.0023
Epoch 28/100
14/14 [==============================] - 0s 2ms/step - loss: -1664877.1250 - acc: 0.0023
Epoch 29/100
14/14 [==============================] - 0s 2ms/step - loss: -1777085.0000 - acc: 0.0023
Epoch 30/100
14/14 [==============================] - 0s 2ms/step - loss: -1885909.3750 - acc: 0.0023
```

```
Epoch 31/100
14/14 [==============================] - 0s 2ms/step - loss: -2007551.7500 - acc: 0.0023
Epoch 32/100
14/14 [==============================] - 0s 2ms/step - loss: -2131553.7500 - acc: 0.0023
Epoch 33/100
14/14 [==============================] - 0s 2ms/step - loss: -2259071.2500 - acc: 0.0023
Epoch 34/100
14/14 [==============================] - 0s 2ms/step - loss: -2391481.2500 - acc: 0.0023
Epoch 35/100
14/14 [==============================] - 0s 2ms/step - loss: -2529461.2500 - acc: 0.0023
Epoch 36/100
14/14 [==============================] - 0s 2ms/step - loss: -2671705.2500 - acc: 0.0023
Epoch 37/100
14/14 [==============================] - 0s 2ms/step - loss: -2816845.7500 - acc: 0.0023
Epoch 38/100
14/14 [==============================] - 0s 2ms/step - loss: -2968668.5000 - acc: 0.0023
Epoch 39/100
14/14 [==============================] - 0s 2ms/step - loss: -3122646.7500 - acc: 0.0023
Epoch 40/100
14/14 [==============================] - 0s 2ms/step - loss: -3280551.7500 - acc: 0.0023
Epoch 41/100
14/14 [==============================] - 0s 2ms/step - loss: -3443374.7500 - acc: 0.0023
Epoch 42/100
14/14 [==============================] - 0s 2ms/step - loss: -3609876.5000 - acc: 0.0023
Epoch 43/100
14/14 [==============================] - 0s 2ms/step - loss: -3779442.7500 - acc: 0.0023
Epoch 44/100
14/14 [==============================] - 0s 2ms/step - loss: -3955006.5000 - acc: 0.0023
Epoch 45/100
14/14 [==============================] - 0s 2ms/step - loss: -4128241.5000 - acc: 0.0023
```

```
[ ] Epoch 46/100
14/14 [==============================] - 0s 2ms/step - loss: -4314072.5000 - acc: 0.0023
Epoch 47/100
14/14 [==============================] - 0s 2ms/step - loss: -4500277.0000 - acc: 0.0023
Epoch 48/100
14/14 [==============================] - 0s 2ms/step - loss: -4691510.5000 - acc: 0.0023
Epoch 49/100
14/14 [==============================] - 0s 3ms/step - loss: -4887914.5000 - acc: 0.0023
Epoch 50/100
14/14 [==============================] - 0s 2ms/step - loss: -5084834.5000 - acc: 0.0023
Epoch 51/100
14/14 [==============================] - 0s 2ms/step - loss: -5282867.5000 - acc: 0.0023
Epoch 52/100
14/14 [==============================] - 0s 2ms/step - loss: -5491495.5000 - acc: 0.0023
Epoch 53/100
14/14 [==============================] - 0s 2ms/step - loss: -5700808.0000 - acc: 0.0023
Epoch 54/100
14/14 [==============================] - 0s 2ms/step - loss: -5911285.0000 - acc: 0.0023
Epoch 55/100
14/14 [==============================] - 0s 2ms/step - loss: -6130830.5000 - acc: 0.0023
Epoch 56/100
14/14 [==============================] - 0s 2ms/step - loss: -6353125.5000 - acc: 0.0023
Epoch 57/100
14/14 [==============================] - 0s 2ms/step - loss: -6580479.5000 - acc: 0.0023
Epoch 58/100
14/14 [==============================] - 0s 2ms/step - loss: -6813616.0000 - acc: 0.0023
Epoch 59/100
14/14 [==============================] - 0s 2ms/step - loss: -7043495.5000 - acc: 0.0023
Epoch 60/100
14/14 [==============================] - 0s 2ms/step - loss: -7283459.5000 - acc: 0.0023
```

```
Epoch 60/100
14/14 [==============================] - 0s 2ms/step - loss: -7283459.5000 - acc: 0.0023
Epoch 61/100
14/14 [==============================] - 0s 2ms/step - loss: -7524759.5000 - acc: 0.0023
Epoch 62/100
14/14 [==============================] - 0s 2ms/step - loss: -7758576.0000 - acc: 0.0023
Epoch 63/100
14/14 [==============================] - 0s 2ms/step - loss: -8006701.0000 - acc: 0.0023
Epoch 64/100
14/14 [==============================] - 0s 2ms/step - loss: -8251303.0000 - acc: 0.0023
Epoch 65/100
14/14 [==============================] - 0s 2ms/step - loss: -8509396.0000 - acc: 0.0023
Epoch 66/100
14/14 [==============================] - 0s 2ms/step - loss: -8759841.0000 - acc: 0.0023
Epoch 67/100
14/14 [==============================] - 0s 2ms/step - loss: -9016615.0000 - acc: 0.0023
Epoch 68/100
14/14 [==============================] - 0s 2ms/step - loss: -9282350.0000 - acc: 0.0023
Epoch 69/100
14/14 [==============================] - 0s 2ms/step - loss: -9536389.0000 - acc: 0.0023
Epoch 70/100
14/14 [==============================] - 0s 2ms/step - loss: -9804974.0000 - acc: 0.0023
Epoch 71/100
14/14 [==============================] - 0s 2ms/step - loss: -10072459.0000 - acc: 0.0023
Epoch 72/100
14/14 [==============================] - 0s 2ms/step - loss: -10345549.0000 - acc: 0.0023
Epoch 73/100
14/14 [==============================] - 0s 2ms/step - loss: -10618057.0000 - acc: 0.0023
Epoch 74/100
14/14 [==============================] - 0s 2ms/step - loss: -10903332.0000 - acc: 0.0023
```

```
Epoch 75/100
14/14 [==============================] - 0s 2ms/step - loss: -11191310.0000 - acc: 0.0023
Epoch 76/100
14/14 [==============================] - 0s 2ms/step - loss: -11480384.0000 - acc: 0.0023
Epoch 77/100
14/14 [==============================] - 0s 3ms/step - loss: -11770965.0000 - acc: 0.0023
Epoch 78/100
14/14 [==============================] - 0s 2ms/step - loss: -12059689.0000 - acc: 0.0023
Epoch 79/100
14/14 [==============================] - 0s 2ms/step - loss: -12359122.0000 - acc: 0.0023
Epoch 80/100
14/14 [==============================] - 0s 2ms/step - loss: -12659065.0000 - acc: 0.0023
Epoch 81/100
14/14 [==============================] - 0s 2ms/step - loss: -12962568.0000 - acc: 0.0023
Epoch 82/100
14/14 [==============================] - 0s 2ms/step - loss: -13269379.0000 - acc: 0.0023
Epoch 83/100
14/14 [==============================] - 0s 2ms/step - loss: -13582745.0000 - acc: 0.0023
Epoch 84/100
14/14 [==============================] - 0s 2ms/step - loss: -13883934.0000 - acc: 0.0023
Epoch 85/100
14/14 [==============================] - 0s 2ms/step - loss: -14202501.0000 - acc: 0.0023
Epoch 86/100
14/14 [==============================] - 0s 2ms/step - loss: -14521539.0000 - acc: 0.0023
Epoch 87/100
14/14 [==============================] - 0s 2ms/step - loss: -14847483.0000 - acc: 0.0023
Epoch 88/100
14/14 [==============================] - 0s 2ms/step - loss: -15167232.0000 - acc: 0.0023
Epoch 89/100
14/14 [==============================] - 0s 2ms/step - loss: -15492096.0000 - acc: 0.0023
```

```
Epoch 90/100
14/14 [==============================] - 0s 2ms/step - loss: -15823391.0000 - acc: 0.0023
Epoch 91/100
14/14 [==============================] - 0s 2ms/step - loss: -16157030.0000 - acc: 0.0023
Epoch 92/100
14/14 [==============================] - 0s 2ms/step - loss: -16499512.0000 - acc: 0.0023
Epoch 93/100
14/14 [==============================] - 0s 2ms/step - loss: -16839516.0000 - acc: 0.0023
Epoch 94/100
14/14 [==============================] - 0s 2ms/step - loss: -17185826.0000 - acc: 0.0023
Epoch 95/100
14/14 [==============================] - 0s 3ms/step - loss: -17540576.0000 - acc: 0.0023
Epoch 96/100
14/14 [==============================] - 0s 2ms/step - loss: -17892166.0000 - acc: 0.0023
Epoch 97/100
14/14 [==============================] - 0s 2ms/step - loss: -18252736.0000 - acc: 0.0023
Epoch 98/100
14/14 [==============================] - 0s 2ms/step - loss: -18611252.0000 - acc: 0.0023
Epoch 99/100
14/14 [==============================] - 0s 2ms/step - loss: -18978914.0000 - acc: 0.0023
Epoch 100/100
14/14 [==============================] - 0s 2ms/step - loss: -19349180.0000 - acc: 0.0023
Model Summary:
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_8 (Dense)             (None, 20)                600

 dense_9 (Dense)             (None, 1)                 21
```

```
=================================================================
Total params: 621 (2.43 KB)
Trainable params: 621 (2.43 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
5/5 [==============================] - 0s 3ms/step - loss: -18084532.0000 - acc: 0.0000e+00

Evaluation Result (loss, accuracy): [-18084532.0, 0.0]
```

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()

```python
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.preprocessing import LabelEncoder, StandardScaler

# Load dataset
dataset = pd.read_csv("Breas Cancer.csv")
dataset.drop(['id'], axis=1, inplace=True)
del dataset['Unnamed: 32']

# Separate features and target variable
X = dataset.iloc[:, 2:].values
Y = dataset.iloc[:, 1].values

# Encode target variable
label = LabelEncoder()
Y = label.fit_transform(Y)

# Split dataset into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25)

# Standardize features
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
# Standardize features
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Set random seed for reproducibility
np.random.seed(155)

# Define the architecture of the neural network model
my_first_nn = Sequential()  # create model
my_first_nn.add(Dense(20, input_dim=29, activation='relu'))  # hidden layer
my_first_nn.add(Dense(1, activation='sigmoid'))  # output layer

# Compile the model
my_first_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])

# Train the model
my_first_nn_fitted = my_first_nn.fit(X_train, Y_train, epochs=100, initial_epoch=0)

# Print model summary
print("Model Summary:")
print(my_first_nn.summary())

# Evaluate the model on the test set
evaluation_result = my_first_nn.evaluate(X_test, Y_test)
print("\nEvaluation Result (loss, accuracy):", evaluation_result)
```

OUTPUT:-

```
Epoch 1/100
14/14 [==============================] - 1s 2ms/step - loss: 21.5153 - acc: 0.0047
Epoch 2/100
14/14 [==============================] - 0s 2ms/step - loss: -97.3750 - acc: 0.0047
Epoch 3/100
14/14 [==============================] - 0s 2ms/step - loss: -215.9197 - acc: 0.0047
Epoch 4/100
14/14 [==============================] - 0s 2ms/step - loss: -339.8655 - acc: 0.0047
Epoch 5/100
14/14 [==============================] - 0s 2ms/step - loss: -471.6372 - acc: 0.0047
Epoch 6/100
14/14 [==============================] - 0s 2ms/step - loss: -615.2157 - acc: 0.0047
Epoch 7/100
14/14 [==============================] - 0s 2ms/step - loss: -771.7296 - acc: 0.0023
Epoch 8/100
14/14 [==============================] - 0s 2ms/step - loss: -942.5661 - acc: 0.0023
Epoch 9/100
14/14 [==============================] - 0s 2ms/step - loss: -1132.9773 - acc: 0.0023
Epoch 10/100
14/14 [==============================] - 0s 2ms/step - loss: -1336.4514 - acc: 0.0023
Epoch 11/100
14/14 [==============================] - 0s 2ms/step - loss: -1560.4012 - acc: 0.0023
Epoch 12/100
14/14 [==============================] - 0s 2ms/step - loss: -1809.7042 - acc: 0.0023
Epoch 13/100
14/14 [==============================] - 0s 2ms/step - loss: -2066.8425 - acc: 0.0023
Epoch 14/100
14/14 [==============================] - 0s 2ms/step - loss: -2353.4475 - acc: 0.0023
Epoch 15/100
14/14 [==============================] - 0s 2ms/step - loss: -2654.7258 - acc: 0.0023
```

```
Epoch 16/100
14/14 [==============================] - 0s 2ms/step - loss: -2987.3862 - acc: 0.0023
Epoch 17/100
14/14 [==============================] - 0s 2ms/step - loss: -3331.4580 - acc: 0.0023
Epoch 18/100
14/14 [==============================] - 0s 3ms/step - loss: -3698.0842 - acc: 0.0023
Epoch 19/100
14/14 [==============================] - 0s 2ms/step - loss: -4101.8672 - acc: 0.0023
Epoch 20/100
14/14 [==============================] - 0s 2ms/step - loss: -4503.1069 - acc: 0.0023
Epoch 21/100
14/14 [==============================] - 0s 2ms/step - loss: -4969.5146 - acc: 0.0023
Epoch 22/100
14/14 [==============================] - 0s 2ms/step - loss: -5424.3398 - acc: 0.0023
Epoch 23/100
14/14 [==============================] - 0s 2ms/step - loss: -5954.6074 - acc: 0.0023
Epoch 24/100
14/14 [==============================] - 0s 2ms/step - loss: -6486.3887 - acc: 0.0023
Epoch 25/100
14/14 [==============================] - 0s 2ms/step - loss: -7084.1865 - acc: 0.0023
Epoch 26/100
14/14 [==============================] - 0s 2ms/step - loss: -7708.9355 - acc: 0.0023
Epoch 27/100
14/14 [==============================] - 0s 2ms/step - loss: -8354.2256 - acc: 0.0023
Epoch 28/100
14/14 [==============================] - 0s 2ms/step - loss: -9041.5361 - acc: 0.0023
Epoch 29/100
14/14 [==============================] - 0s 2ms/step - loss: -9743.8115 - acc: 0.0023
Epoch 30/100
14/14 [==============================] - 0s 2ms/step - loss: -10522.3662 - acc: 0.0023
```

```
Epoch 31/100
14/14 [==============================] - 0s 2ms/step - loss: -11279.3271 - acc: 0.0023
Epoch 32/100
14/14 [==============================] - 0s 2ms/step - loss: -12071.5449 - acc: 0.0023
Epoch 33/100
14/14 [==============================] - 0s 2ms/step - loss: -12895.2793 - acc: 0.0023
Epoch 34/100
14/14 [==============================] - 0s 2ms/step - loss: -13735.8838 - acc: 0.0023
Epoch 35/100
14/14 [==============================] - 0s 2ms/step - loss: -14623.9424 - acc: 0.0023
Epoch 36/100
14/14 [==============================] - 0s 2ms/step - loss: -15530.2383 - acc: 0.0023
Epoch 37/100
14/14 [==============================] - 0s 2ms/step - loss: -16480.9102 - acc: 0.0023
Epoch 38/100
14/14 [==============================] - 0s 2ms/step - loss: -17440.1152 - acc: 0.0023
Epoch 39/100
14/14 [==============================] - 0s 2ms/step - loss: -18407.2617 - acc: 0.0023
Epoch 40/100
14/14 [==============================] - 0s 2ms/step - loss: -19413.8125 - acc: 0.0023
Epoch 41/100
14/14 [==============================] - 0s 2ms/step - loss: -20492.0312 - acc: 0.0023
Epoch 42/100
14/14 [==============================] - 0s 2ms/step - loss: -21524.0938 - acc: 0.0023
Epoch 43/100
14/14 [==============================] - 0s 3ms/step - loss: -22653.0820 - acc: 0.0023
Epoch 44/100
14/14 [==============================] - 0s 2ms/step - loss: -23798.5742 - acc: 0.0023
Epoch 45/100
14/14 [==============================] - 0s 2ms/step - loss: -24916.7539 - acc: 0.0023
Epoch 46/100
```

```
Epoch 46/100
14/14 [==============================] - 0s 2ms/step - loss: -26133.1016 - acc: 0.0023
Epoch 47/100
14/14 [==============================] - 0s 2ms/step - loss: -27307.8477 - acc: 0.0023
Epoch 48/100
14/14 [==============================] - 0s 2ms/step - loss: -28587.1777 - acc: 0.0023
Epoch 49/100
14/14 [==============================] - 0s 2ms/step - loss: -29801.4160 - acc: 0.0023
Epoch 50/100
14/14 [==============================] - 0s 2ms/step - loss: -31093.0078 - acc: 0.0023
Epoch 51/100
14/14 [==============================] - 0s 2ms/step - loss: -32368.3516 - acc: 0.0023
Epoch 52/100
14/14 [==============================] - 0s 2ms/step - loss: -33605.7031 - acc: 0.0023
Epoch 53/100
14/14 [==============================] - 0s 2ms/step - loss: -34945.9805 - acc: 0.0023
Epoch 54/100
14/14 [==============================] - 0s 2ms/step - loss: -36347.7852 - acc: 0.0023
Epoch 55/100
14/14 [==============================] - 0s 2ms/step - loss: -37727.2344 - acc: 0.0023
Epoch 56/100
14/14 [==============================] - 0s 2ms/step - loss: -39094.0391 - acc: 0.0023
Epoch 57/100
14/14 [==============================] - 0s 2ms/step - loss: -40562.3438 - acc: 0.0023
Epoch 58/100
14/14 [==============================] - 0s 2ms/step - loss: -41972.5898 - acc: 0.0023
Epoch 59/100
14/14 [==============================] - 0s 3ms/step - loss: -43463.8164 - acc: 0.0023
Epoch 60/100
14/14 [==============================] - 0s 2ms/step - loss: -44897.4648 - acc: 0.0023
Epoch 61/100
```

```
14/14 [==============================] - 0s 2ms/step - loss: -46407.6211 - acc: 0.0023
Epoch 62/100
14/14 [==============================] - 0s 2ms/step - loss: -47953.0859 - acc: 0.0023
Epoch 63/100
14/14 [==============================] - 0s 2ms/step - loss: -49551.0078 - acc: 0.0023
Epoch 64/100
14/14 [==============================] - 0s 2ms/step - loss: -51155.8359 - acc: 0.0023
Epoch 65/100
14/14 [==============================] - 0s 2ms/step - loss: -52743.7070 - acc: 0.0023
Epoch 66/100
14/14 [==============================] - 0s 2ms/step - loss: -54413.0742 - acc: 0.0023
Epoch 67/100
14/14 [==============================] - 0s 2ms/step - loss: -56017.6992 - acc: 0.0023
Epoch 68/100
14/14 [==============================] - 0s 2ms/step - loss: -57598.1953 - acc: 0.0023
Epoch 69/100
14/14 [==============================] - 0s 2ms/step - loss: -59367.5781 - acc: 0.0023
Epoch 70/100
14/14 [==============================] - 0s 2ms/step - loss: -61058.7812 - acc: 0.0023
Epoch 71/100
14/14 [==============================] - 0s 2ms/step - loss: -62811.6055 - acc: 0.0023
Epoch 72/100
14/14 [==============================] - 0s 2ms/step - loss: -64571.7227 - acc: 0.0023
Epoch 73/100
14/14 [==============================] - 0s 2ms/step - loss: -66376.9141 - acc: 0.0023
Epoch 74/100
14/14 [==============================] - 0s 2ms/step - loss: -68312.2422 - acc: 0.0023
Epoch 75/100
14/14 [==============================] - 0s 4ms/step - loss: -70092.6719 - acc: 0.0023
Epoch 76/100
```

```
Epoch 95/100
14/14 [==============================] - 0s 2ms/step - loss: -110424.5000 - acc: 0.0023
Epoch 96/100
14/14 [==============================] - 0s 3ms/step - loss: -112704.5078 - acc: 0.0023
Epoch 97/100
14/14 [==============================] - 0s 3ms/step - loss: -115031.1953 - acc: 0.0023
Epoch 98/100
14/14 [==============================] - 0s 3ms/step - loss: -117312.5938 - acc: 0.0023
Epoch 99/100
14/14 [==============================] - 0s 3ms/step - loss: -119582.2734 - acc: 0.0023
Epoch 100/100
14/14 [==============================] - 0s 3ms/step - loss: -121850.7734 - acc: 0.0023
Model Summary:
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_10 (Dense)            (None, 20)                600

 dense_11 (Dense)            (None, 1)                 21

=================================================================
Total params: 621 (2.43 KB)
Trainable params: 621 (2.43 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
5/5 [==============================] - 0s 3ms/step - loss: -116084.9688 - acc: 0.0000e+00

Evaluation Result (loss, accuracy): [-116084.96875, 0.0]
```

In class programming:

2. Use Image Classification on the hand written digits data set (mnist)

```
from keras import Sequential
from keras.datasets import mnist
import numpy as np
from keras.layers import Dense
from keras.utils import to_categorical

(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

print(train_images.shape[1:])
# process the data
# 1. convert each image of shape 28*28 to 784 dimensional which will be fed to the network as a single feature
dimData = np.prod(train_images.shape[1:])
print(dimData)
train_data = train_images.reshape(train_images.shape[0], dimData)
test_data = test_images.reshape(test_images.shape[0], dimData)

# convert data to float and scale values between 0 and 1
train_data = train_data.astype('float') / 255.0
test_data = test_data.astype('float') / 255.0

# change the labels from integer to one-hot encoding. to_categorical is doing the same thing as LabelEncoder()
train_labels_one_hot = to_categorical(train_labels)
test_labels_one_hot = to_categorical(test_labels)

# creating network
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(dimData,)))
```

```
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(dimData,)))
model.add(Dense(512, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(train_data, train_labels_one_hot, batch_size=256, epochs=10, verbose=1,
                    validation_data=(test_data, test_labels_one_hot))
```

OUTPUT:-

```
(28, 28)
784
Epoch 1/10
235/235 [==============================] - 7s 25ms/step - loss: 0.2921 - accuracy: 0.9103 - val_loss: 0.1437 -
Epoch 2/10
235/235 [==============================] - 6s 26ms/step - loss: 0.0993 - accuracy: 0.9700 - val_loss: 0.1174 -
Epoch 3/10
235/235 [==============================] - 5s 21ms/step - loss: 0.0626 - accuracy: 0.9804 - val_loss: 0.0823 -
Epoch 4/10
235/235 [==============================] - 6s 24ms/step - loss: 0.0428 - accuracy: 0.9862 - val_loss: 0.0635 -
Epoch 5/10
235/235 [==============================] - 5s 23ms/step - loss: 0.0313 - accuracy: 0.9899 - val_loss: 0.0847 -
Epoch 6/10
235/235 [==============================] - 5s 21ms/step - loss: 0.0226 - accuracy: 0.9932 - val_loss: 0.0598 -
Epoch 7/10
235/235 [==============================] - 6s 27ms/step - loss: 0.0163 - accuracy: 0.9948 - val_loss: 0.0791 -
Epoch 8/10
235/235 [==============================] - 5s 21ms/step - loss: 0.0120 - accuracy: 0.9963 - val_loss: 0.0670 -
Epoch 9/10
235/235 [==============================] - 6s 26ms/step - loss: 0.0097 - accuracy: 0.9969 - val_loss: 0.0780 -
Epoch 10/10
235/235 [==============================] - 5s 21ms/step - loss: 0.0074 - accuracy: 0.9977 - val_loss: 0.0818 -
```

1. Plot the loss and accuracy for both training data and validation data using the history object in the source code.

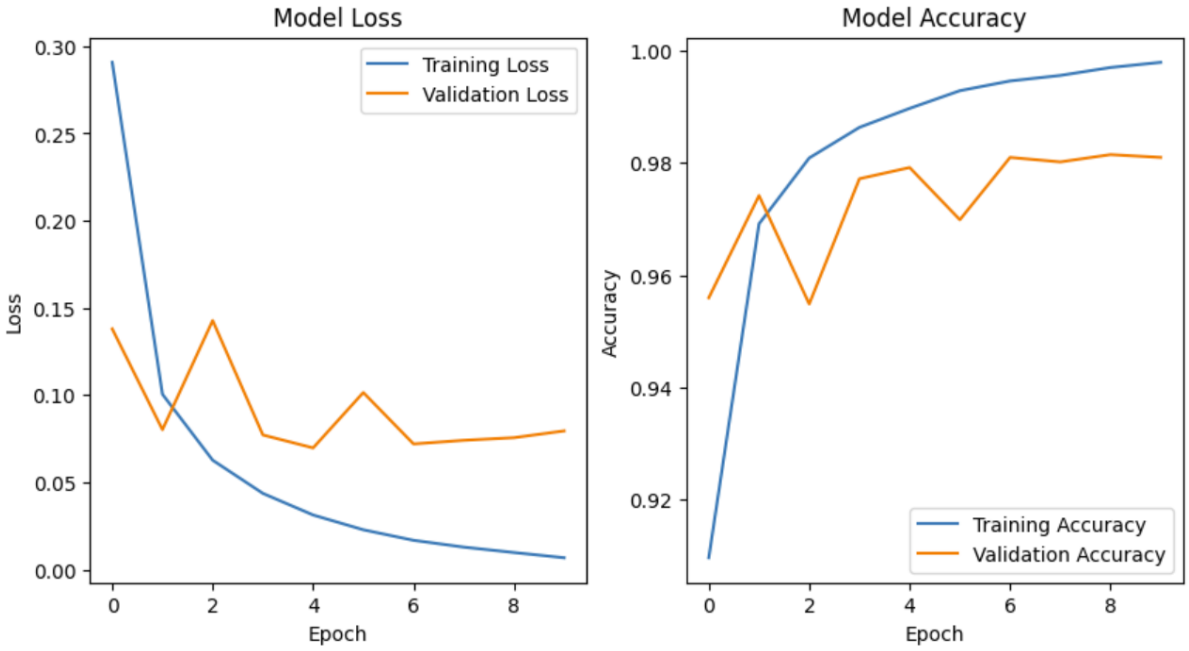
```
from keras import Sequential
from keras.datasets import mnist
import numpy as np
from keras.layers import Dense
from keras.utils import to_categorical

(train_images,train_labels),(test_images, test_labels) = mnist.load_data()

print(train_images.shape[1:])
#process the data
#1. convert each image of shape 28*28 to 784 dimensional which will be fed to the network as a single feature
dimData = np.prod(train_images.shape[1:])
print(dimData)
train_data = train_images.reshape(train_images.shape[0],dimData)
test_data = test_images.reshape(test_images.shape[0],dimData)

#convert data to float and scale values between 0 and 1
train_data = train_data.astype('float')
test_data = test_data.astype('float')
#scale data
train_data /=255.0
test_data /=255.0
#change the labels frominteger to one-hot encoding. to_categorical is doing the same thing as LabelEncoder()
train_labels_one_hot = to_categorical(train_labels)
test_labels_one_hot = to_categorical(test_labels)
```

```
#change the labels frominteger to one-hot encoding. to_categorical is doing the same thing as LabelEncoder()
train_labels_one_hot = to_categorical(train_labels)
test_labels_one_hot = to_categorical(test_labels)

#creating network
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(dimData,)))
model.add(Dense(512, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(train_data, train_labels_one_hot, batch_size=256, epochs=10, verbose=1,
                    validation_data=(test_data, test_labels_one_hot))

import matplotlib.pyplot as plt

# Plot training & validation loss values
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

# Plot training & validation accuracy values
plt.subplot(1, 2, 2)
```

```
# Plot training & validation accuracy values
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()


#model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
#history = model.fit(train_data, train_labels_one_hot, batch_size=256, epochs=10, verbose=1,
 #                   validation_data=(test_data, test_labels_one_hot))
```

OUTPUT:-

```
(28, 28)
784
Epoch 1/10
235/235 [==============================] - 7s 27ms/step - loss: 0.2908 - accuracy: 0.9097 - val_loss: 0.1380 -
Epoch 2/10
235/235 [==============================] - 5s 22ms/step - loss: 0.1005 - accuracy: 0.9692 - val_loss: 0.0802 -
Epoch 3/10
235/235 [==============================] - 5s 23ms/step - loss: 0.0627 - accuracy: 0.9809 - val_loss: 0.1427 -
Epoch 4/10
235/235 [==============================] - 6s 24ms/step - loss: 0.0436 - accuracy: 0.9863 - val_loss: 0.0771 -
Epoch 5/10
235/235 [==============================] - 5s 23ms/step - loss: 0.0313 - accuracy: 0.9897 - val_loss: 0.0697 -
Epoch 6/10
235/235 [==============================] - 6s 26ms/step - loss: 0.0228 - accuracy: 0.9929 - val_loss: 0.1015 -
Epoch 7/10
235/235 [==============================] - 5s 23ms/step - loss: 0.0167 - accuracy: 0.9946 - val_loss: 0.0720 -
Epoch 8/10
235/235 [==============================] - 7s 28ms/step - loss: 0.0128 - accuracy: 0.9956 - val_loss: 0.0741 -
Epoch 9/10
235/235 [==============================] - 5s 21ms/step - loss: 0.0097 - accuracy: 0.9970 - val_loss: 0.0756 -
Epoch 10/10
235/235 [==============================] - 6s 26ms/step - loss: 0.0067 - accuracy: 0.9979 - val_loss: 0.0795 -
```



2. Plot one of the images in the test data, and then do inferencing to check what is the prediction of the model on that single image.

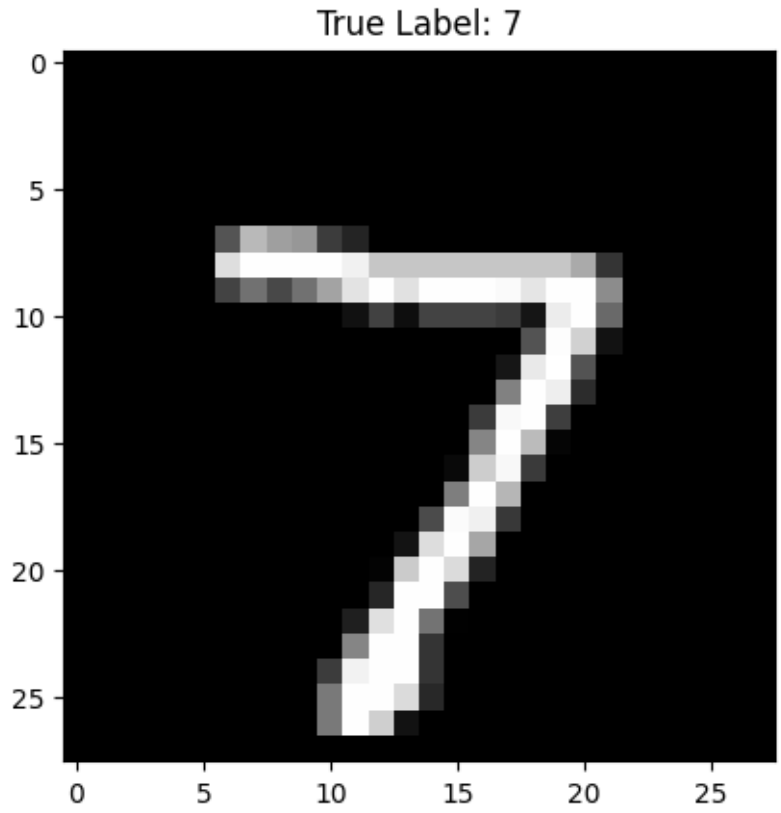
```
from keras import Sequential
from keras.datasets import mnist
import numpy as np
from keras.layers import Dense
from keras.utils import to_categorical

# Load the MNIST dataset
(train_images,train_labels),(test_images, test_labels) = mnist.load_data()

# Print the shape of the training images
print(train_images.shape[1:])

# Process the data
# Convert each image of shape 28*28 to 784 dimensional, which will be fed to the network as a single feature
dimData = np.prod(train_images.shape[1:])
print(dimData)
train_data = train_images.reshape(train_images.shape[0],dimData)
test_data = test_images.reshape(test_images.shape[0],dimData)

# Convert data to float and scale values between 0 and 1
train_data = train_data.astype('float')
test_data = test_data.astype('float')
# Scale data
train_data /=255.0
test_data /=255.0

# Change the labels from integer to one-hot encoding
```

```
# Change the labels from integer to one-hot encoding
train_labels_one_hot = to_categorical(train_labels)
test_labels_one_hot = to_categorical(test_labels)

# Creating the neural network model
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(dimData,)))
model.add(Dense(512, activation='relu'))
model.add(Dense(10, activation='softmax'))

# Compile the model
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
history = model.fit(train_data, train_labels_one_hot, batch_size=256, epochs=10, verbose=1,
                    validation_data=(test_data, test_labels_one_hot))

import matplotlib.pyplot as plt

# Plot one image from the test data
plt.figure()
plt.imshow(test_images[0], cmap='gray')
plt.title(f"True Label: {test_labels[0]}")

# Perform inference on the single image
image = test_data[0].reshape(1, dimData)
prediction = model.predict(image)
predicted_label = np.argmax(prediction)
```

```
# Perform inference on the single image
image = test_data[0].reshape(1, dimData)
prediction = model.predict(image)
predicted_label = np.argmax(prediction)

print(f"Predicted Label: {predicted_label}")
```

OUTPUT:-

```
11490434/11490434 [==============================] - 1s 0us/step
(28, 28)
784
Epoch 1/10
235/235 [==============================] - 8s 29ms/step - loss: 0.2910 - accuracy: 0.9110 - val_loss: 0.1282 -
Epoch 2/10
235/235 [==============================] - 5s 23ms/step - loss: 0.0997 - accuracy: 0.9696 - val_loss: 0.0919 -
Epoch 3/10
235/235 [==============================] - 7s 28ms/step - loss: 0.0637 - accuracy: 0.9803 - val_loss: 0.0709 -
Epoch 4/10
235/235 [==============================] - 5s 23ms/step - loss: 0.0432 - accuracy: 0.9860 - val_loss: 0.0720 -
Epoch 5/10
235/235 [==============================] - 7s 28ms/step - loss: 0.0309 - accuracy: 0.9903 - val_loss: 0.0664 -
Epoch 6/10
235/235 [==============================] - 5s 23ms/step - loss: 0.0223 - accuracy: 0.9929 - val_loss: 0.0941 -
Epoch 7/10
235/235 [==============================] - 6s 25ms/step - loss: 0.0167 - accuracy: 0.9948 - val_loss: 0.0637 -
Epoch 8/10
235/235 [==============================] - 6s 26ms/step - loss: 0.0128 - accuracy: 0.9958 - val_loss: 0.0609 -
Epoch 9/10
235/235 [==============================] - 6s 25ms/step - loss: 0.0090 - accuracy: 0.9973 - val_loss: 0.0681 -
Epoch 10/10
235/235 [==============================] - 6s 27ms/step - loss: 0.0062 - accuracy: 0.9982 - val_loss: 0.0620 -
1/1 [==============================] - 0s 81ms/step
Predicted Label: 7
```



3. We had used 2 hidden layers and Relu activation. Try to change the number of hidden layer and the activation to tanh or sigmoid and see what happens.

```
from keras import Sequential
from keras.datasets import mnist
import numpy as np
from keras.layers import Dense
from keras.utils import to_categorical

# Load the MNIST dataset
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

# Print the shape of the training images
print(train_images.shape[1:])

# Process the data
# 1. Convert each image of shape 28*28 to 784 dimensional, which will be fed to the network as a single featur
dimData = np.prod(train_images.shape[1:])
print(dimData)
train_data = train_images.reshape(train_images.shape[0], dimData)
test_data = test_images.reshape(test_images.shape[0], dimData)

# Convert data to float and scale values between 0 and 1
train_data = train_data.astype('float')
test_data = test_data.astype('float')
# Scale data
train_data /= 255.0
test_data /= 255.0

# Change the labels from integer to one-hot encoding
train_labels_one_hot = to_categorical(train_labels)
test_labels_one_hot = to_categorical(test_labels)

# Creating the neural network model
model = Sequential()
model.add(Dense(512, activation='tanh', input_shape=(dimData,)))  # Changed activation to 'tanh'
model.add(Dense(256, activation='tanh'))  # Added another hidden layer with 'tanh'
model.add(Dense(128, activation='tanh'))  # Added one more hidden layer with 'tanh'
model.add(Dense(10, activation='softmax'))

# Compile the model
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
history = model.fit(train_data, train_labels_one_hot, batch_size=256, epochs=10, verbose=1,
                    validation_data=(test_data, test_labels_one_hot))
```

OUTPUT:-

```
(28, 28)
784
Epoch 1/10
235/235 [==============================] - 9s 27ms/step - loss: 0.3394 - accuracy: 0.8962 - val_loss: 0.1729 -
Epoch 2/10
235/235 [==============================] - 5s 20ms/step - loss: 0.1449 - accuracy: 0.9570 - val_loss: 0.1157 -
Epoch 3/10
235/235 [==============================] - 6s 25ms/step - loss: 0.0964 - accuracy: 0.9715 - val_loss: 0.0932 -
Epoch 4/10
235/235 [==============================] - 5s 20ms/step - loss: 0.0702 - accuracy: 0.9783 - val_loss: 0.0906 -
Epoch 5/10
235/235 [==============================] - 5s 20ms/step - loss: 0.0519 - accuracy: 0.9837 - val_loss: 0.0769 -
Epoch 6/10
235/235 [==============================] - 6s 25ms/step - loss: 0.0399 - accuracy: 0.9876 - val_loss: 0.0738 -
Epoch 7/10
235/235 [==============================] - 5s 21ms/step - loss: 0.0308 - accuracy: 0.9904 - val_loss: 0.0841 -
Epoch 8/10
235/235 [==============================] - 6s 25ms/step - loss: 0.0227 - accuracy: 0.9933 - val_loss: 0.0718 -
Epoch 9/10
235/235 [==============================] - 5s 20ms/step - loss: 0.0165 - accuracy: 0.9951 - val_loss: 0.0732 -
Epoch 10/10
235/235 [==============================] - 5s 21ms/step - loss: 0.0129 - accuracy: 0.9965 - val_loss: 0.0638 -
```

4. Run the same code without scaling the images and check the performance?

```
from keras import Sequential
from keras.datasets import mnist
import numpy as np
from keras.layers import Dense
from keras.utils import to_categorical

(train_images,train_labels),(test_images, test_labels) = mnist.load_data()

print(train_images.shape[1:])
#process the data
#1. convert each image of shape 28*28 to 784 dimensional which will be fed to the network as a single feature
dimData = np.prod(train_images.shape[1:])
print(dimData)
train_data = train_images.reshape(train_images.shape[0],dimData)
test_data = test_images.reshape(test_images.shape[0],dimData)

#convert data to float and scale values between 0 and 1
train_data = train_data.astype('float')
test_data = test_data.astype('float')
#scale data
#train_data /=255.0
#test_data /=255.0
#change the labels frominteger to one-hot encoding. to_categorical is doing the same thing as LabelEncoder()
train_labels_one_hot = to_categorical(train_labels)
test_labels_one_hot = to_categorical(test_labels)

#creating network
model = Sequential()
```

```
train_labels_one_hot = to_categorical(train_labels)
test_labels_one_hot = to_categorical(test_labels)

#creating network
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(dimData,)))
model.add(Dense(512, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(train_data, train_labels_one_hot, batch_size=256, epochs=10, verbose=1,
                    validation_data=(test_data, test_labels_one_hot))
```

OUTPUT:-

```
(28, 28)
784
Epoch 1/10
235/235 [==============================] - 10s 40ms/step - loss: 5.6614 - accuracy: 0.8777 - val_loss: 0.7007 -
Epoch 2/10
235/235 [==============================] - 8s 34ms/step - loss: 0.4260 - accuracy: 0.9466 - val_loss: 0.4194 -
Epoch 3/10
235/235 [==============================] - 9s 36ms/step - loss: 0.2534 - accuracy: 0.9596 - val_loss: 0.3492 -
Epoch 4/10
235/235 [==============================] - 9s 37ms/step - loss: 0.2018 - accuracy: 0.9676 - val_loss: 0.5652 -
Epoch 5/10
235/235 [==============================] - 9s 36ms/step - loss: 0.1605 - accuracy: 0.9740 - val_loss: 0.2677 -
Epoch 6/10
235/235 [==============================] - 7s 32ms/step - loss: 0.1411 - accuracy: 0.9765 - val_loss: 0.3155 -
Epoch 7/10
235/235 [==============================] - 7s 29ms/step - loss: 0.1408 - accuracy: 0.9785 - val_loss: 0.2825 -
Epoch 8/10
235/235 [==============================] - 7s 31ms/step - loss: 0.1209 - accuracy: 0.9826 - val_loss: 0.3198 -
Epoch 9/10
235/235 [==============================] - 7s 30ms/step - loss: 0.1142 - accuracy: 0.9835 - val_loss: 0.3881 -
Epoch 10/10
235/235 [==============================] - 7s 28ms/step - loss: 0.1169 - accuracy: 0.9851 - val_loss: 0.3564 -
```