

Ex No: 2**BUILD A SIMPLE NEURAL NETWORKS****AIM:**

To build a simple neural network using Keras/TensorFlow.

PROCEDURE:

1. Download and load the dataset.
2. Perform analysis and preprocessing of the dataset.
3. Build a simple neural network model using Keras/TensorFlow.
4. Compile and fit the model.
5. Perform prediction with the test dataset.
6. Calculate performance metrics.

PROGRAM:

```
# first neural network with keras make predictions
from numpy import loadtxt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# load the dataset
dataset = loadtxt('pima-indians-diabetes.csv', delimiter=',')

# split into input (X) and output (y) variables
X = dataset[:,0:8]
y = dataset[:,8]

# define the keras model
model = Sequential()
model.add(Dense(12, input_shape=(8,), activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

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

# fit the keras model on the dataset
model.fit(X, y, epochs=150, batch_size=10, verbose=0)
```

```
# make class predictions with the model

predictions = (model.predict(X) > 0.5).astype(int)

# summarize the first 5 cases

for i in range(5):

    print('%s => %d (expected %d)' % (X[i].tolist(), predictions[i], y[i]))
```

OUTPUT:

```
[1]: # first neural network with keras tutorial
from numpy import loadtxt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

[3]: # Load the dataset
dataset = loadtxt('pima-indian-diabetes.csv', delimiter=',')
# split into input (X) and output (y) variables
X = dataset[:,0:8]
y = dataset[:,8]

[6]: # define the keras model
model = Sequential()
model.add(Dense(12, input_shape=(8,), activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

[7]: model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

[8]: # fit the keras model on the dataset
model.fit(X, y, epochs=150, batch_size=10)

Epoch 143/150
77/77 [=====] - 0s 1ms/step - loss: 0.4854 - accuracy: 0.7721
Epoch 144/150
77/77 [=====] - 0s 1ms/step - loss: 0.4880 - accuracy: 0.7760
Epoch 145/150
77/77 [=====] - 0s 1ms/step - loss: 0.4933 - accuracy: 0.7604
Epoch 146/150
77/77 [=====] - 0s 1ms/step - loss: 0.4728 - accuracy: 0.7826
Epoch 147/150
77/77 [=====] - 0s 1ms/step - loss: 0.4923 - accuracy: 0.7734
Epoch 148/150
77/77 [=====] - 0s 1ms/step - loss: 0.4836 - accuracy: 0.7773
Epoch 149/150
77/77 [=====] - 0s 1ms/step - loss: 0.4802 - accuracy: 0.7669
Epoch 150/150
77/77 [=====] - 0s 1ms/step - loss: 0.4901 - accuracy: 0.7721

[8]: <keras.callbacks.History at 0x1d0ae27aec0>

[9]: _, accuracy = model.evaluate(X, y)
print('Accuracy: %.2f' % (accuracy*100))

24/24 [=====] - 0s 1ms/step - loss: 0.4636 - accuracy: 0.7865
Accuracy: 78.65
```

Result:

A simple neural network using Keras/TensorFlow is successfully build.