5. Deep Learning – First Neural Network with Keras

Contents

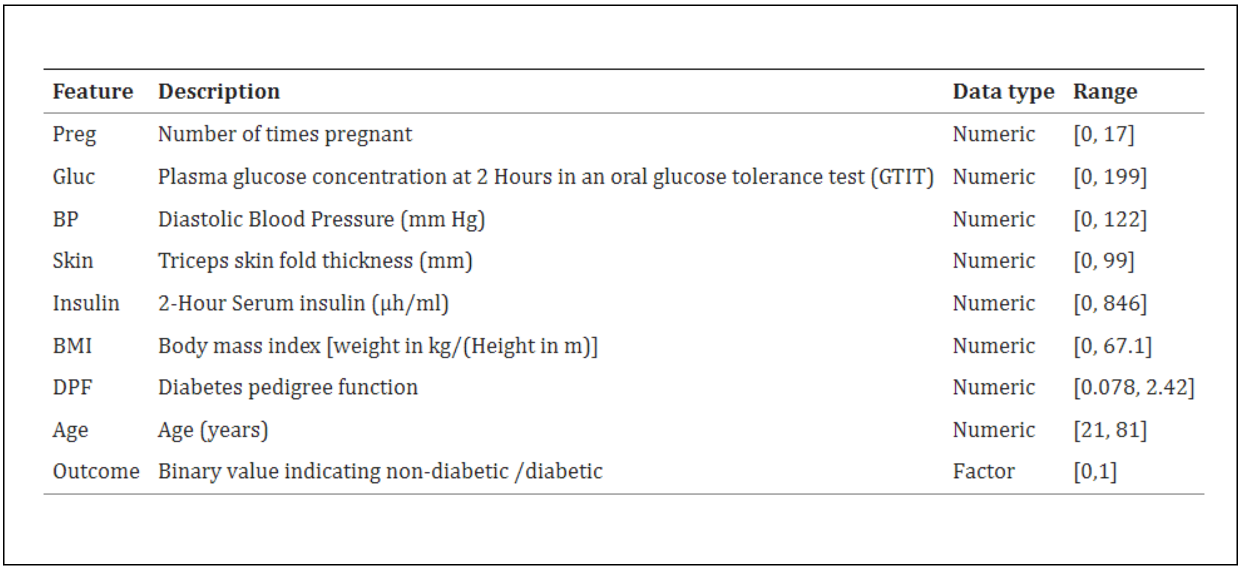
5. Deep Learning – First Neural Network with Keras

1. Implementing first neural network using keras

- ✓ Importing the libraries
- ✓ Loading Dataset
- ✓ Data preparation
- ✓ Splitting the dataset
- ✓ Model creation
- ✓ Model compilation
- ✓ Model training
- ✓ Prediction

2. Dataset explanation

- ✓ We are going to work with **pima-indians-diabetes.csv**
- ✓ This Dataset is related to health care domain.
- ✓ Pima Indians are a Native American group that lives in Mexico and Arizona, USA.
- ✓ It describes patient medical record data for Pima Indians and whether they had a diabetes within five years.
- ✓ The Pima Indian Diabetes dataset consisting of Pima Indian females 21 years and older is a popular benchmark dataset.
- ✓ It is a binary classification problem (onset of diabetes as 1 or not as 0).
- ✓ All of the input variables that describe each patient are numerical.



Feature	Description	Data type	Range
Preg	Number of times pregnant	Numeric	[0, 17]
Gluc	Plasma glucose concentration at 2 Hours in an oral glucose tolerance test (GTIT)	Numeric	[0, 199]
BP	Diastolic Blood Pressure (mm Hg)	Numeric	[0, 122]
Skin	Triceps skin fold thickness (mm)	Numeric	[0, 99]
Insulin	2-Hour Serum insulin (µh/ml)	Numeric	[0, 846]
BMI	Body mass index [weight in kg/(Height in m)]	Numeric	[0, 67.1]
DPF	Diabetes pedigree function	Numeric	[0.078, 2.42]
Age	Age (years)	Numeric	[21, 81]
Outcome	Binary value indicating non-diabetic /diabetic	Factor	[0,1]

3. Input and output from the Dataset

3.1. Input Variables (X):

- ✓ 1. Number of times pregnant
- ✓ 2. Plasma glucose concentration at 2 hours in an oral glucose tolerance test
- ✓ 3. Diastolic blood pressure (mm Hg)
- ✓ 4. Triceps skin fold thickness (mm)
- ✓ 5. 2-hour serum insulin (μIU/ml)
- ✓ 6. Body mass index (weight in kg/(height in m))
- ✓ 7. Diabetes pedigree function
- ✓ 8. Age (years)

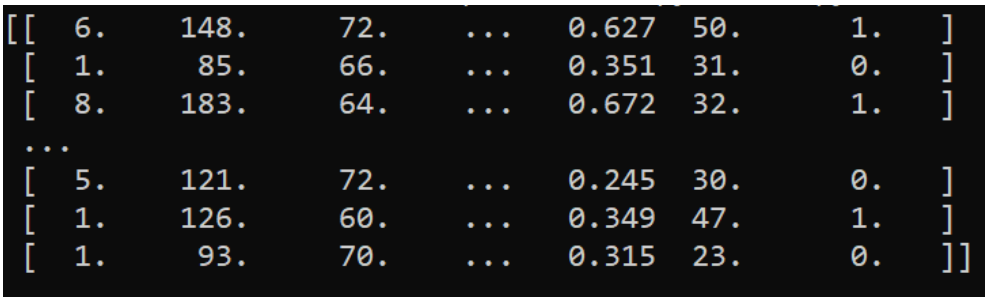
3.2. Output Variables (y):

- ✓ 1. Class variable (0 or 1)

Program Loading csv file
Name demo1.py
Input file pima-indians-diabetes.csv

```
from numpy import loadtxt

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

print(dataset)
```

Output



```
[[  6.     148.      72.     ...   0.627  50.        1.    ]
 [  1.      85.      66.     ...   0.351  31.        0.    ]
 [  8.     183.      64.     ...   0.672  32.        1.    ]
 ...
 [  5.     121.      72.     ...   0.245  30.        0.    ]
 [  1.     126.      60.     ...   0.349  47.        1.    ]
 [  1.      93.      70.     ...   0.315  23.        0.    ]]
```

Program Split into input (X) and output (y) variables
Name demo2.py
Input file pima-indians-diabetes.csv

```
from numpy import loadtxt

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

X = dataset[:, 0:8]
y = dataset[:, 8]

print("Splitting data done")
```

Output

```
Splitting data done
```

4. Create the model

- ✓ First step is, we need to create the model by using Sequential class
- ✓ Once model created then we need to add layers to the model

Program Model creation
Name demo4.py
Input file pima-indians-diabetes.csv

```
from numpy import loadtxt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

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

X = dataset[:, 0:8]
y = dataset[:, 8]

model = Sequential()

layer1 = Dense(12, input_shape = (8, ), activation = 'relu')
layer2 = Dense(8, activation = 'relu')
layer3 = Dense(1, activation = 'sigmoid')

model.add(layer1)
model.add(layer2)
model.add(layer3)

print("Model created")
```

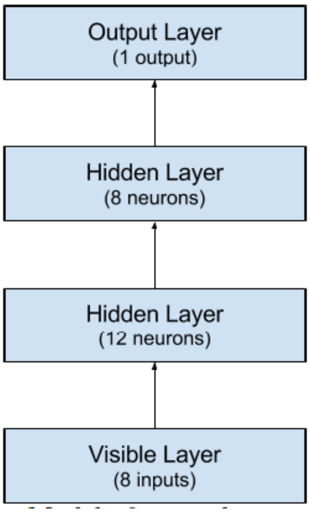
Output

```
Model created
```

4.1. Input shape and activation functions

- ✓ Fully connected layers are defined using the Dense class.
- ✓ The model expects rows of data with **8** features (the input_shape = (**8**,) argument).
- ✓ The first hidden layer has **12** nodes and uses the relu activation function.
- ✓ The second hidden layer has **8** nodes and uses the relu activation function.
- ✓ The output layer has **1** node and uses the sigmoid activation function.

- ✓ The line of code, Dense layer is doing two things, creating first hidden and input layers.



5. Compile the keras model

- ✓ Once the model created then we need to compile the model

Program Compile the keras model
Name demo5.py
Input file pima-indians-diabetes.csv

```
# importing required libraries
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()

layer1 = Dense(12, input_shape = (8, ), activation = 'relu')
layer2 = Dense(8, activation = 'relu')
layer3 = Dense(1, activation = 'sigmoid')

model.add(layer1)
model.add(layer2)
model.add(layer3)

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

print("Model compiled")
```

Output

```
Model created
```

5.1. Loss function, optimiser and metrics

- ✓ We need to provide loss function to evaluate a set of weights.
- ✓ The optimizer is used to search through different weights for the network.
- ✓ This loss is for a binary classification problems and is defined in Keras as “binary_crossentropy”.
- ✓ We have given optimizer value as adam.
  - This is more efficient gradient descent algorithm.

6. Fit the model

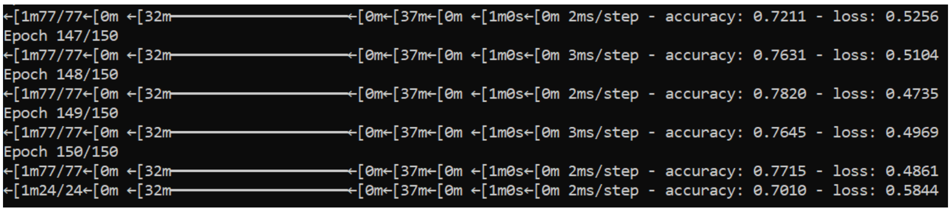
- ✓ Once model compiled then we need to train the model
- ✓ By using fit(…) method we can train the model.

Program	Fit the model
Name	demo6.py
Input file	pima-indians-diabetes.csv

```
print("Topic: First DL Example")
print()


print("Step 1: Importing libraries")

from numpy import loadtxt
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential

import warnings
warnings.filterwarnings("ignore")



print("Step 2: Loading the dataset")

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


print("Step 3: Data preparation")

X = dataset[:, 0:8]
y = dataset[:, 8]
```

```
print("Step 4: Splitting the dataset: Optional")

print("Step 5: Model(Neural Network) creation")

model = Sequential()


print("Step 5.1: Creating layers and add to the model")

layers12 = Dense(12, input_shape = (8, ), activation = 'relu')
layer3 = Dense(8, activation = 'relu')
layer4 = Dense(1, activation = 'sigmoid')

model.add(layers12)
model.add(layer3)
model.add(layer4)


print("Step 6: Model compilation")

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

print("Step 7: Model training")

model.fit(
   X,
   y,
   epochs = 150,
   batch_size = 10,
)
```

Output



```
←[1m77/77←[0m ←[32m━━━━━━━━━━━━━━━━━━━━←[0m←[37m←[0m ←[1m0s←[0m 2ms/step - accuracy: 0.7211 - loss: 0.5256
Epoch 147/150
←[1m77/77←[0m ←[32m━━━━━━━━━━━━━━━━━━━━←[0m←[37m←[0m ←[1m0s←[0m 3ms/step - accuracy: 0.7631 - loss: 0.5104
Epoch 148/150
←[1m77/77←[0m ←[32m━━━━━━━━━━━━━━━━━━━━←[0m←[37m←[0m ←[1m0s←[0m 2ms/step - accuracy: 0.7820 - loss: 0.4735
Epoch 149/150
←[1m77/77←[0m ←[32m━━━━━━━━━━━━━━━━━━━━←[0m←[37m←[0m ←[1m0s←[0m 3ms/step - accuracy: 0.7645 - loss: 0.4969
Epoch 150/150
←[1m77/77←[0m ←[32m━━━━━━━━━━━━━━━━━━━━←[0m←[37m←[0m ←[1m0s←[0m 2ms/step - accuracy: 0.7715 - loss: 0.4861
←[1m24/24←[0m ←[32m━━━━━━━━━━━━━━━━━━━━←[0m←[37m←[0m ←[1m0s←[0m 2ms/step - accuracy: 0.7010 - loss: 0.5844
```

7. Evaluate the model

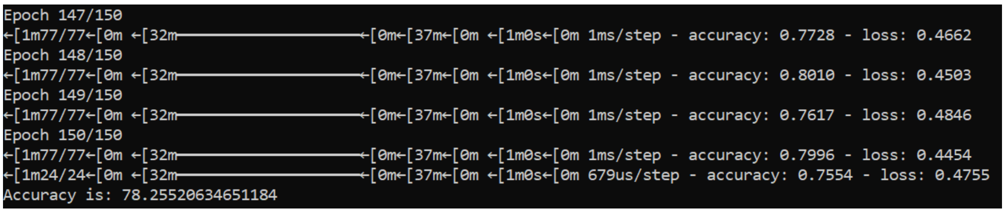
- ✓ Once training is done then we need to evaluate the performance of the network.
- ✓ By using evaluate() method we can evaluate.
- ✓ This method return two values,
    - ○ The first will be the loss of the model on the dataset.
    - ○ The second will be the accuracy of the model on the dataset.

Program Name	Fit the model demo7.py
Input file	pima-indians-diabetes.csv

```
print("Topic: First DL Example")
print()

print("Step 1: Importing libraries")

from numpy import loadtxt
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential

import warnings
warnings.filterwarnings("ignore")


print("Step 2: Loading the dataset")

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


print("Step 3: Data preparation")

X = dataset[:, 0:8]
y = dataset[:, 8]
```

```
print("Step 4: Splitting the dataset: Optional")

print("Step 5: Model(Neural Network) creation")

model = Sequential()

print("Step 5.1: Creating layers and add to the model")

layers12 = Dense(12, input_shape = (8, ), activation = 'relu')
layer3 = Dense(8, activation = 'relu')
layer4 = Dense(1, activation = 'sigmoid')

model.add(layers12)
model.add(layer3)
model.add(layer4)


print("Step 6: Model compilation")

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

print("Step 7: Model training")

model.fit(
    X,
    y,
    epochs = 150,
    batch_size = 10,
)

# Evaluate the keras model

_, accuracy = model.evaluate(X, y)
print("Accuracy is:", accuracy*100)
```

Output



```
Epoch 147/150
←[1m77/77←[0m ←[32m━━━━━━━━━━━━━━━━━━━━←[0m←[37m←[0m ←[1m0s←[0m 1ms/step - accuracy: 0.7728 - loss: 0.4662
Epoch 148/150
←[1m77/77←[0m ←[32m━━━━━━━━━━━━━━━━━━━━←[0m←[37m←[0m ←[1m0s←[0m 1ms/step - accuracy: 0.8010 - loss: 0.4503
Epoch 149/150
←[1m77/77←[0m ←[32m━━━━━━━━━━━━━━━━━━━━←[0m←[37m←[0m ←[1m0s←[0m 1ms/step - accuracy: 0.7617 - loss: 0.4846
Epoch 150/150
←[1m77/77←[0m ←[32m━━━━━━━━━━━━━━━━━━━━←[0m←[37m←[0m ←[1m0s←[0m 1ms/step - accuracy: 0.7996 - loss: 0.4454
←[1m24/24←[0m ←[32m━━━━━━━━━━━━━━━━━━━━←[0m←[37m←[0m ←[1m0s←[0m 679us/step - accuracy: 0.7554 - loss: 0.4755
Accuracy is: 78.25520634651184
```

8. Prediction

- ✓ By using predict(…) method we can do the prediction.
- ✓ We are using sigmoid activation function on the output layer.
    - The predictions will be a probability in the range between 0 and 1

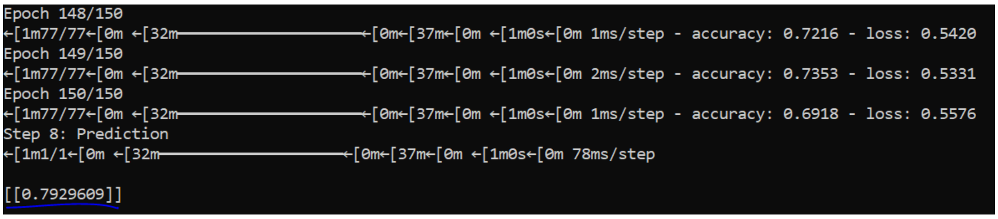
Program	Model prediction
Name	demo8.py
Input file	pima-indians-diabetes.csv

```
print("Topic: First DL Example")
print()

print("Step 1: Importing libraries")

import numpy as np
from numpy import loadtxt
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential

import warnings
warnings.filterwarnings("ignore")


print("Step 2: Loading the dataset")

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


print("Step 3: Data preparation")

X = dataset[:, 0:8]
y = dataset[:, 8]
```

```
print("Step 4: Splitting the dataset: Optional")

print("Step 5: Model(Neural Network) creation")

model = Sequential()


print("Step 5.1: Creating layers and add to the model")

layers12 = Dense(12, input_shape = (8, ), activation = 'relu')
layer3 = Dense(8, activation = 'relu')
layer4 = Dense(1, activation = 'sigmoid')

model.add(layers12)
model.add(layer3)
model.add(layer4)


print("Step 6: Model compilation")

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

print("Step 7: Model training")

model.fit(
   X,
   y,
   epochs = 150,
   batch_size = 10,
)
```

```
print("Step 8: Prediction")

ip = [[6, 148, 72, 35, 0, 33.6, 0.627, 50]]

input_data = np.array(ip)
result = model.predict(input_data)

print()
print(result)
```

Output



```
Epoch 148/150
←[1m77/77←[0m ←[32m━━━━━━━━━━━━━━━━━━━━←[0m←[37m←[0m ←[1m0s←[0m 1ms/step - accuracy: 0.7216 - loss: 0.5420
Epoch 149/150
←[1m77/77←[0m ←[32m━━━━━━━━━━━━━━━━━━━━←[0m←[37m←[0m ←[1m0s←[0m 2ms/step - accuracy: 0.7353 - loss: 0.5331
Epoch 150/150
←[1m77/77←[0m ←[32m━━━━━━━━━━━━━━━━━━━━←[0m←[37m←[0m ←[1m0s←[0m 1ms/step - accuracy: 0.6918 - loss: 0.5576
Step 8: Prediction
←[1m1/1←[0m ←[32m━━━━━━━━━━━━━━━━━━━━←[0m←[37m←[0m ←[1m0s←[0m 78ms/step

[[0.7929609]]
```

Note

- ✓ Here model done prediction for first five rows compared to the expected class value.
- ✓ You can see that most rows are correctly predicted.
- ✓ We got 79.2% accuracy with good model performance.

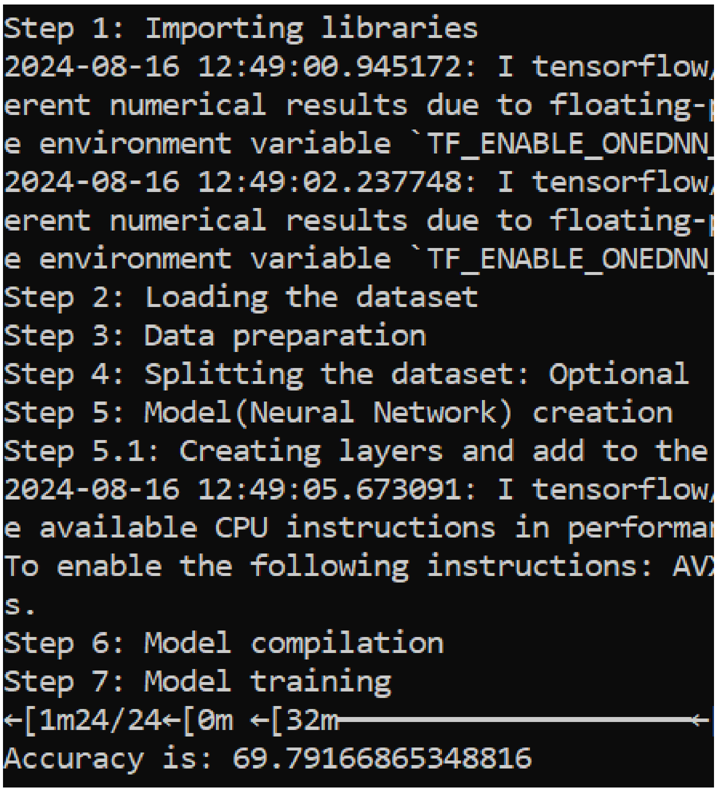
9. verbose = 0

- ✓ If we provide verbose = 0 then progress bar will be not displayed.

Program	Model prediction
Name	demo9.py
Input file	pima-indians-diabetes.csv

```
print("Topic: First DL Example")
print()

print("Step 1: Importing libraries")

from numpy import loadtxt
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential

import warnings
warnings.filterwarnings("ignore")


print("Step 2: Loading the dataset")

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

print("Step 3: Data preparation")

X = dataset[:, 0:8]
y = dataset[:, 8]


print("Step 4: Splitting the dataset: Optional")
print("Step 5: Model(Neural Network) creation")

model = Sequential()


print("Step 5.1: Creating layers and add to the model")

layers12 = Dense(12, input_shape = (8, ), activation = 'relu')
layer3 = Dense(8, activation = 'relu')
layer4 = Dense(1, activation = 'sigmoid')

model.add(layers12)
model.add(layer3)
model.add(layer4)


print("Step 6: Model compilation")

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

print("Step 7: Model training")

model.fit(
   X,
   y,
   epochs = 150,
   batch_size = 10,
   verbose = 0

)

# Evaluate the keras model

_, accuracy = model.evaluate(X, y)
print("Accuracy is:", accuracy*100)
```

Output



```
Step 1: Importing libraries
2024-08-16 12:49:00.945172: I tensorflow/
erent numerical results due to floating-p
e environment variable `TF_ENABLE_ONEDNN_
2024-08-16 12:49:02.237748: I tensorflow/
erent numerical results due to floating-p
e environment variable `TF_ENABLE_ONEDNN_
Step 2: Loading the dataset
Step 3: Data preparation
Step 4: Splitting the dataset: Optional
Step 5: Model(Neural Network) creation
Step 5.1: Creating layers and add to the
2024-08-16 12:49:05.673091: I tensorflow/
e available CPU instructions in performan
To enable the following instructions: AVX
s.
Step 6: Model compilation
Step 7: Model training
←[1m24/24←[0m ←[32m━━━━━━━━━━━━━━━━━━━━←
Accuracy is: 69.79166865348816
```

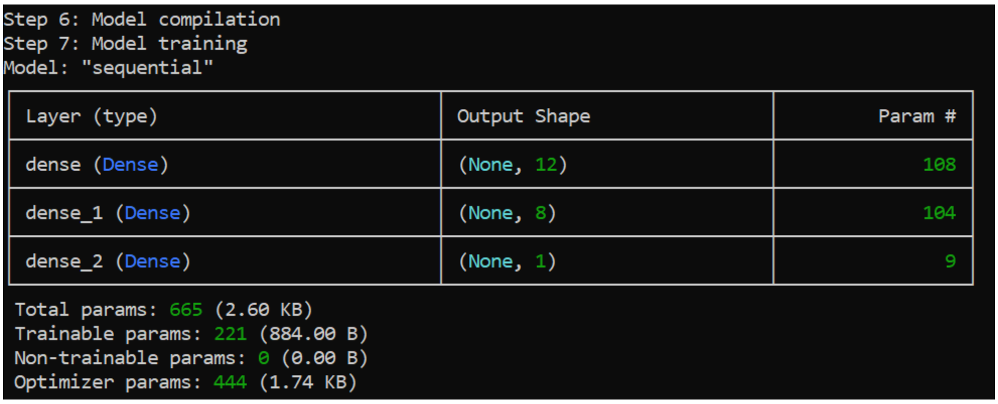
10. Model summary

✓ By using summary method, we can check the summary of the model.

Program Name	Model summary demo10.py
Input file	pima-indians-diabetes.csv

```
print("Topic: First DL Example")
print()

print("Step 1: Importing libraries")

from numpy import loadtxt
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential

import warnings
warnings.filterwarnings("ignore")


print("Step 2: Loading the dataset")

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


print("Step 3: Data preparation")

X = dataset[:, 0:8]
y = dataset[:, 8]

print("Step 4: Splitting the dataset: Optional")
print("Step 5: Model(Neural Network) creation")

model = Sequential()


print("Step 5.1: Creating layers and add to the model")

layers12 = Dense(12, input_shape = (8, ), activation = 'relu')
layer3 = Dense(8, activation = 'relu')
layer4 = Dense(1, activation = 'sigmoid')

model.add(layers12)
model.add(layer3)
model.add(layer4)


print("Step 6: Model compilation")

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

print("Step 7: Model training")

model.fit(
   X,
   y,
   epochs = 150,
   batch_size = 10,
   verbose = 0

)

# Model Summary

model.summary()
```

Output



```
Step 6: Model compilation
Step 7: Model training
Model: "sequential"
┌─────────────────────────────────┬────────────────────────┬───────────────┐
│ Layer (type)                    │ Output Shape           │       Param # │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ (None, 12)             │           108 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                 │ (None, 8)              │           104 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_2 (Dense)                 │ (None, 1)              │             9 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 665 (2.60 KB)
 Trainable params: 221 (884.00 B)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 444 (1.74 KB)
```

11. Image of the model

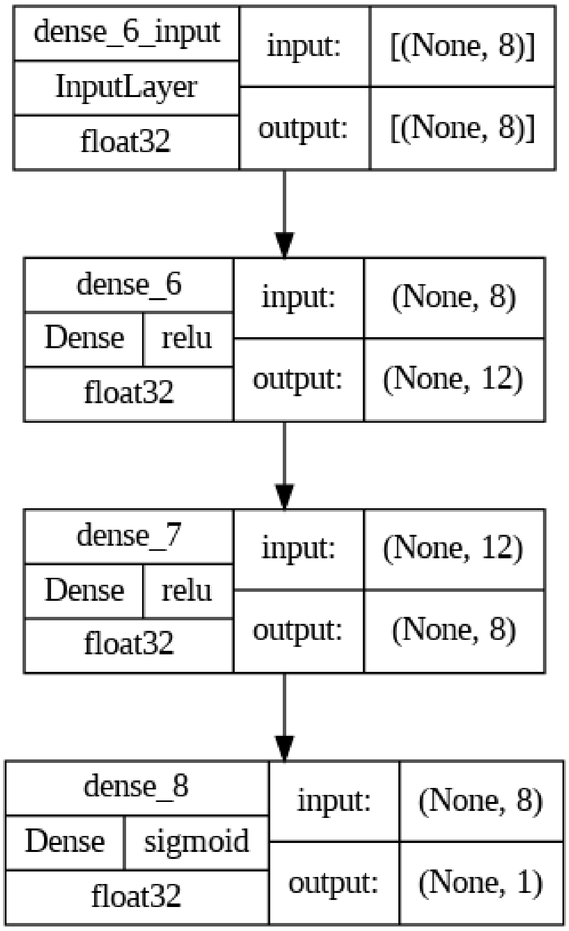
✓ We can get the image of the model

Program Image of the model
Name demo11.py
Input file pima-indians-diabetes.csv

```
# importing required libraries
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()

layer1 = Dense(12, input_shape = (8, ), activation = 'relu')
layer2 = Dense(8, activation = 'relu')
layer3 = Dense(1, activation = 'sigmoid')

model.add(layer1)
model.add(layer2)
model.add(layer3)

# 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)
```

```
from tensorflow.keras.utils import plot_model

plot_model(model, to_file = 'model.png', show_shapes = True,
show_dtype = True, show_layer_names = True, expand_nested =
True, show_layer_activations = True)
```

Output


dense_6_input
InputLayer
float32
input:
[(None, 8)]
output:
[(None, 8)]
dense_6
Dense
relu
float32
input:
(None, 8)
output:
(None, 12)
dense_7
Dense
relu
float32
input:
(None, 12)
output:
(None, 8)
dense_8
Dense
sigmoid
float32
input:
(None, 8)
output:
(None, 1)