In [1]:

```
!pip install tensorflow
# import
import tensorflow as tf
import numpy as np
a = tf.constant(15)
b = tf.constant(20)
print(a+b)
# input
x = np.random.rand(100).astype(np.float32)
#print(x)
# output - observed
y = x * 0.2 + 0.2
# Weight
W = tf.Variable(tf.random.normal([1]))
# bias
b = tf.Variable(tf.zeros([1]))
# Create a function for MSE - mean squared error
def mse_loss():
    ypred = W * x + b
    loss = tf.reduce_mean(tf.square(ypred-y))
    return loss
# Optimizer
optimizer = tf.keras.optimizers.Adam()
# Iterations
for step in range(5000):
    optimizer.minimize(mse_loss,var_list=[W,b])
    if step % 500 == 0:
        print(step, W.numpy(), b.numpy())
#print(mse_loss)
```

```
8.4)
Requirement already satisfied: google-auth<3,>=1.6.3 in c:\users\ananyapranav\appd
ata\roaming\python\python311\site-packages (from tensorboard<2.14,>=2.13->tensorfl
ow-intel==2.13.0->tensorflow) (2.22.0)
Requirement already satisfied: google-auth-oauthlib<1.1,>=0.5 in c:\users\ananyapr
anav\appdata\roaming\python\python311\site-packages (from tensorboard<2.14,>=2.13-
>tensorflow-intel==2.13.0->tensorflow) (1.0.0)
Requirement already satisfied: markdown>=2.6.8 in c:\programdata\anaconda3\lib\sit
e-packages (from tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow)
(3.4.1)
Requirement already satisfied: requests<3,>=2.21.0 in c:\programdata\anaconda3\lib
\site-packages (from tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflo
w) (2.31.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in c:\users\a
nanyapranav\appdata\roaming\python\python311\site-packages (from tensorboard<2.14,
>=2.13->tensorflow-intel==2.13.0->tensorflow) (0.7.1)
Requirement already satisfied: werkzeug>=1.0.1 in c:\programdata\anaconda3\lib\sit
e-packages (from tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow)
(2.2.3)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in c:\users\ananyapranav\app
data\roaming\python\python311\site-packages (from google-auth<3,>=1.6.3->tensorboa
rd<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (5.3.1)
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\programdata\anaconda3\l
ib\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.14,>=2.13->tensorflow-
intel==2.13.0->tensorflow) (0.2.8)
Requirement already satisfied: rsa<5,>=3.1.4 in c:\users\ananyapranav\appdata\roam
ing\python\python311\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.14,>
=2.13->tensorflow-intel==2.13.0->tensorflow) (4.9)
Requirement already satisfied: urllib3<2.0 in c:\programdata\anaconda3\lib\site-pa
ckages (from google-auth<3,>=1.6.3->tensorboard<2.14,>=2.13->tensorflow-intel==2.1
3.0->tensorflow) (1.26.16)
Requirement already satisfied: requests-oauthlib>=0.7.0 in c:\users\ananyapranav\a
ppdata\roaming\python\python311\site-packages (from google-auth-oauthlib<1.1,>=0.5
->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (1.3.1)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\programdata\anaconda
3\lib\site-packages (from requests<3,>=2.21.0->tensorboard<2.14,>=2.13->tensorflow
-intel==2.13.0->tensorflow) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\programdata\anaconda3\lib\site-p
ackages (from requests<3,>=2.21.0->tensorboard<2.14,>=2.13->tensorflow-intel==2.1
3.0->tensorflow) (3.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\programdata\anaconda3\lib
\site-packages (from requests<3,>=2.21.0->tensorboard<2.14,>=2.13->tensorflow-inte
l==2.13.0->tensorflow) (2023.7.22)
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\programdata\anaconda3\lib\s
ite-packages (from werkzeug>=1.0.1->tensorboard<2.14,>=2.13->tensorflow-intel==2.1
3.0->tensorflow) (2.1.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in c:\programdata\anaconda3\li
b\site-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.
14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in c:\users\ananyapranav\appdata\ro
aming\python\python311\site-packages (from requests-oauthlib>=0.7.0->google-auth-o
authlib<1.1,>=0.5->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow)
(3.2.2)
tf.Tensor(35, shape=(), dtype=int32)
0 [-0.74691105] [0.00099999]
500 [-0.3731104] [0.34804583]
1000 [-0.21300863] [0.40408438]
1500 [-0.1105561] [0.36122796]
2000 [-0.01404603] [0.31117767]
2500 [0.06945173] [0.2677748]
3000 [0.13174799] [0.23542407]
3500 [0.1706939] [0.21520887]
4000 [0.19023393] [0.20506814]
4500 [0.19765516] [0.20121685]
```

In [20]:

```
# Step 1 - Load the dataset
from numpy import loadtxt
import pandas as pd
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
## INPUT Variables ##
# x1 - Number of times pregnant
# x2 - plasma glucose
# x3 - diastolic blood pressure
# x4 - Triceps skin fold thickness
# x5 - 2-hour serum insulin
# x6 - bmi
# x7 - diabetes pedigree function
# x8 - age (yrs)
## Output Variable ##
# Class Variable - 0 or 1
dataset = pd.read_csv("pima-indians-diabetes.csv")
dataset.head()
# [:,:] - first : is range of rows and second : is columns
# [start:end] - begins at start, ends at end-1
x = dataset.iloc[:,0:8]
print(type(x))
print(x.shape)
y = dataset.iloc[:,8]
print(y)
# Step 2 - Creating or define the Keras Model
# Sequential Model
# Layer1 -> Layer2 -> Layer3
model = Sequential()
# The model expects row of data with 8 variables
# 12 = nodes
model.add(Dense(12, input_shape=(8,), activation='relu'))
# Hidden Layer
# 8 = nodes
model.add(Dense(8, activation='relu'))
# Output layer
model.add(Dense(1,activation='sigmoid'))
# Step 3 - Compile the Keras model
# loss (error)
# optimizer (adam)
# metrics = accuracy
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
#Step 4 - Fit / Train the model
#1 = Epochs - number of iterations / passes
#2 - Batch - sample data
model.fit(x,y, epochs=150, batch_size=10)
# Step 5 - evaluate the model
model.evaluate(x,y)
```

```
0.7536
Epoch 146/150
77/77 [==============================] - 0s 592us/step - loss: 0.4971 - accuracy:
0.7614
Epoch 147/150
77/77 [==============================] - 0s 605us/step - loss: 0.4843 - accuracy:
0.7653
Epoch 148/150
77/77 [==============================] - 0s 592us/step - loss: 0.4948 - accuracy:
0.7692
Epoch 149/150
77/77 [==============================] - 0s 579us/step - loss: 0.5074 - accuracy:
0.7653
Epoch 150/150
77/77 [==============================] - 0s 579us/step - loss: 0.5169 - accuracy:
0.7640
24/24 [==============================] - 0s 609us/step - loss: 0.4708 - accuracy:
0.7888
```

Out[20]:
```
[0.4708038568496704, 0.7887874841690063]
```

In [18]:
```
dataset.head()
```

Out[18]:

	6	148	72	35	0	33.6	0.627	50	1
0	1	85	66	29	0	26.6	0.351	31	0
1	8	183	64	0	0	23.3	0.672	32	1
2	1	89	66	23	94	28.1	0.167	21	0
3	0	137	40	35	168	43.1	2.288	33	1
4	5	116	74	0	0	25.6	0.201	30	0

In [4]:
```
!pip install torch
```

```
Installing collected packages: torch
Successfully installed torch-2.1.0
```

```
  WARNING: The scripts convert-caffe2-to-onnx.exe, convert-onnx-to-caffe2.exe and
torchrun.exe are installed in 'C:\Users\ANANYAPRANAV\AppData\Roaming\Python\Python
311\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warnin
g, use --no-warn-script-location.
```

In [6]:

```python
 # Two major objectives of PyTorch
#1. Replacement of Numpy to use the power of GPUs and other accelerators
#2. Automatic Differentiation Library helps in implementation of Neural Networks

# Feed Forward --> first stage
# Back Propagation --> second stage (differentiation)
# Tensors - Special Data Structures
# similar to an array, matrices
# similar to Numpy ndarrays
import torch
import numpy as np
# Tensor Initialization
### multiple ways
### 1 - using data
data = [
[1,2],
[3,4]
]
x_data = torch.tensor(data)
print(type(x_data))

### 2 - using numpy array
np_array = np.array(data)
x_np = torch.from_numpy(np_array)
print(x_np)
print(type(x_np))

### 3 - using another tensor
x_ones = torch.ones_like(x_data)
print("One Tensor: \n",x_ones)

x_rand = torch.rand_like(x_data,dtype=torch.float)
print(x_rand)

#### more ways to create tensors
#shape = (2,3)
#random_tensor = torch.rand(shape)
#print(random_tensor)
#print(type(random_tensor))
#ones_tensor = torch.ones(shape)
#print(ones_tensor)
#print(type(ones_tensor))
#zeros_tensor = torch.zeros(shape)
#print(zeros_tensor)
#print(type(zeros_tensor))
#tensor = torch.rand(3,4)
#print(tensor)
#tensor.shape
#tensor.dtype
#tensor.device
# Tensor Operations
#if torch.cuda.is_available():
 #   tensor = tensor.to('cuda')
  #  print("Device tensor is stored in ", tensor.device)
```

```
# Indexing, Slicing
tensor = torch.ones(4,4)
print(tensor)
print(tensor)

tensor1 = torch.zeros(4,4)
print(tensor1)

tensor2 = torch.cat([tensor,tensor1])
print(tensor2)


# Multiply Operation
tensor.mul(tensor1)
tensor * tensor1
tensor.T

# inplace - change the original tensor
tensor.add_(3)
print(tensor)

# from tensor to numpy
t = torch.ones(5)
print(t)

n = t.numpy()
print(n)
print(type(n))
```

```
<class 'torch.Tensor'>
tensor([[1, 2],
        [3, 4]], dtype=torch.int32)
<class 'torch.Tensor'>
One Tensor:
 tensor([[1, 1],
        [1, 1]])
tensor([[0.5079, 0.2607],
        [0.7267, 0.5785]])
tensor([[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.]])
tensor([[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.]])
tensor([[0., 0., 0., 0.],
        [0., 0., 0., 0.],
        [0., 0., 0., 0.],
        [0., 0., 0., 0.]])
tensor([[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [0., 0., 0., 0.],
        [0., 0., 0., 0.],
        [0., 0., 0., 0.],
        [0., 0., 0., 0.]])
tensor([[4., 4., 4., 4.],
        [4., 4., 4., 4.],
        [4., 4., 4., 4.],
        [4., 4., 4., 4.]])
tensor([1., 1., 1., 1., 1.])
[1. 1. 1. 1. 1.]
<class 'numpy.ndarray'>
```