In [88]:

```
import tensorflow as tf
import pandas as pd
from tensorflow.keras import layers
from tensorflow import keras
```

In [89]:

```
mydata=pd.read_csv(r"C:\Users\ANKIT SINGH\Desktop\internship\DataSets-master\500_Person_Gender_Height_Weight_Index.csv")
```

In [90]:

```
mydata[:2]
```

Out[90]:

	Gender	Height	Weight	Index
0	Male	174	96	4
1	Male	189	87	2

In [91]:

```
X=mydata.iloc[:,0:3]
Y=mydata.iloc[:,3]   #output is index
```

```
In [134]: X
```

Out[134]:

	Gender	Height	Weight
0	0	174	96
1	0	189	87
2	1	185	110
3	1	195	104
4	0	149	61
5	0	189	104
6	0	147	92
7	0	154	111
8	0	174	90
9	1	169	103
10	0	195	81
11	1	159	80
12	1	192	101
13	0	155	51
14	0	191	79
15	1	153	107
16	1	157	110
17	0	140	129
18	0	144	145
19	0	172	139
20	0	157	110
21	1	153	149
22	1	169	97
23	0	185	139
24	1	172	67

	Gender	Height	Weight
25	1	151	64
26	0	190	95
27	0	187	62
28	1	163	159
29	0	179	152
...	...	...	...
470	0	147	142
471	0	154	112
472	1	178	65
473	0	195	153
474	1	167	79
475	0	183	131
476	1	164	142
477	0	167	64
478	1	151	55
479	1	147	107
480	1	155	115
481	1	172	108
482	1	142	86
483	0	146	85
484	1	188	115
485	0	173	111
486	1	160	109
487	0	187	80
488	0	198	136
489	1	179	150
490	1	164	59

	Gender	Height	Weight
491	1	146	147
492	1	198	50
493	1	170	53
494	0	152	98
495	1	150	153
496	1	184	121
497	1	141	136
498	0	150	95
499	0	173	131

500 rows × 3 columns

In [135]:

```
Y  #index values separate
```

Out[135]:

```
0      4
1      2
2      4
3      3
4      3
5      3
6      5
7      5
8      3
9      4
10     2
11     4
12     3
13     2
14     2
15     5
16     5
17     5
18     5
19     5
20     5
21     5
22     4
23     5
24     2
25     3
26     3
27     1
28     5
29     5
      ..
470    5
471    5
472    2
473    5
474    3
475    4
476    5
477    2
478    2
```

```
479    5
480    5
481    4
482    5
483    4
484    4
485    4
486    5
487    2
488    4
489    5
490    2
491    5
492    0
493    1
494    5
495    5
496    4
497    5
498    5
499    5
Name: Index, Length: 500, dtype: int64
```

In [137]:

```
Index_name=pd.Series(["Extremely Weak","Weak","Normal","Overweight","Obesity","Extreme Obesity"])
Index_name
```

Out[137]:

```
0     Extremely Weak
1               Weak
2             Normal
3         Overweight
4            Obesity
5    Extreme Obesity
dtype: object
```

In [138]:

```
X["Gender"]=X["Gender"].map({"Male":0,"Female":1})
```

In [94]:

```
XA=X.values
YA=Y.values
```

In [95]:
```
XA
```

Out[95]:
```
array([[  0, 174,  96],
       [  0, 189,  87],
       [  1, 185, 110],
       ...,
       [  1, 141, 136],
       [  0, 150,  95],
       [  0, 173, 131]], dtype=int64)
```

In [96]:
```
YA
```

Out[96]:
```
array([4, 2, 4, 3, 3, 3, 5, 5, 3, 4, 2, 4, 3, 2, 2, 5, 5, 5, 5, 5, 5, 5,
       4, 5, 2, 3, 3, 1, 5, 5, 5, 1, 1, 5, 5, 4, 3, 4, 5, 2, 4, 5, 2, 5,
       4, 2, 4, 4, 3, 5, 5, 1, 5, 4, 4, 3, 4, 5, 3, 5, 0, 5, 0, 2, 5, 5,
       4, 2, 4, 4, 2, 4, 5, 2, 3, 4, 4, 4, 4, 0, 3, 5, 3, 4, 5, 0, 5, 5,
       5, 5, 5, 3, 3, 2, 4, 5, 4, 5, 1, 0, 4, 5, 5, 4, 4, 4, 5, 5, 4, 3,
       4, 5, 4, 2, 4, 3, 2, 5, 5, 5, 4, 4, 4, 5, 5, 4, 4, 4, 5, 2, 5, 2,
       5, 4, 5, 5, 5, 3, 5, 5, 2, 4, 5, 5, 5, 4, 3, 5, 3, 3, 0, 3, 3, 5,
       5, 4, 3, 5, 3, 4, 2, 2, 3, 5, 4, 2, 4, 5, 3, 2, 4, 5, 5, 4, 4, 4,
       4, 3, 5, 3, 3, 4, 4, 2, 3, 3, 5, 3, 5, 4, 5, 5, 4, 5, 5, 5, 4, 4,
       5, 5, 1, 3, 4, 4, 5, 4, 5, 4, 3, 4, 4, 5, 5, 5, 0, 5, 5, 5, 5, 5,
       2, 5, 4, 5, 0, 5, 3, 4, 5, 5, 4, 2, 3, 3, 4, 3, 5, 5, 2, 5, 3, 2,
       1, 5, 0, 5, 3, 5, 3, 4, 3, 5, 5, 5, 5, 2, 4, 5, 5, 4, 5, 5, 5, 2,
       4, 5, 5, 5, 5, 1, 5, 5, 4, 0, 3, 3, 4, 2, 3, 1, 1, 5, 5, 4, 4, 4,
       4, 5, 2, 5, 4, 3, 3, 4, 5, 5, 2, 4, 3, 4, 5, 4, 2, 4, 5, 4, 5, 5,
       1, 5, 5, 5, 5, 2, 2, 5, 3, 5, 4, 5, 4, 4, 5, 5, 4, 2, 2, 4, 3, 3,
       5, 4, 2, 2, 2, 2, 5, 5, 4, 5, 3, 4, 4, 3, 4, 4, 2, 2, 5, 2, 2, 2,
       2, 5, 0, 3, 4, 5, 1, 4, 1, 4, 5, 4, 5, 5, 3, 4, 5, 4, 3, 5, 1, 2,
       4, 5, 5, 5, 5, 3, 5, 1, 4, 5, 5, 2, 5, 4, 3, 2, 2, 2, 2, 3, 5, 3,
       3, 5, 3, 5, 3, 4, 2, 4, 4, 5, 2, 5, 5, 5, 1, 4, 5, 5, 5, 4, 5, 2,
       5, 2, 1, 5, 5, 4, 1, 1, 4, 4, 4, 4, 2, 5, 5, 4, 2, 5, 5, 5, 1, 5,
       4, 2, 5, 5, 4, 5, 4, 4, 5, 5, 5, 4, 5, 0, 2, 2, 4, 2, 4, 5, 4, 5,
       1, 5, 2, 5, 3, 5, 5, 3, 5, 5, 2, 5, 3, 4, 5, 2, 2, 5, 5, 4, 5, 4,
       4, 4, 5, 2, 4, 5, 2, 5, 0, 1, 5, 5, 4, 5, 5, 5], dtype=int64)
```

In [97]:
```
print("Type od XA {} and Type of YA {}".format(type(XA),type(YA)))
```

```
Type od XA <class 'numpy.ndarray'> and Type of YA <class 'numpy.ndarray'>
```

In [98]:
```
Yc=tf.keras.utils.to_categorical(YA,num_classes=6,dtype="float32")
Yc
```

Out[98]:
```
array([[0., 0., 0., 0., 1., 0.],
       [0., 0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0.],
       ...,
       [0., 0., 0., 0., 0., 1.],
       [0., 0., 0., 0., 0., 1.],
       [0., 0., 0., 0., 0., 1.]], dtype=float32)
```

In [99]:
```
type(Yc)
```

Out[99]:
```
numpy.ndarray
```

In [100]:
```
from sklearn.model_selection import train_test_split
```

In [123]:
```
xtrain,xtest,ytrain,ytest=train_test_split(XA,Yc,test_size=.30,random_state=101)
```

In [124]:
```
model=tf.keras.Sequential()
```

In [125]:
```
model.add(layers.Dense(20,input_dim=3,activation="relu"))
model.add(layers.Dense(20,activation="relu"))
model.add(layers.Dense(6,activation='softmax'))

#compiling model
model.compile(loss="categorical_crossentropy",optimizer="adam",metrics=['accuracy'])
```

```
In [126]: model.fit(xtrain,ytrain,epochs=500,batch_size=100)
```

```
Epoch 492/500
350/350 [==============================] - 0s 43us/sample - loss: 0.5694 - acc: 0.7400
Epoch 493/500
350/350 [==============================] - 0s 37us/sample - loss: 0.5619 - acc: 0.7429
Epoch 494/500
350/350 [==============================] - 0s 34us/sample - loss: 0.5609 - acc: 0.7429
Epoch 495/500
350/350 [==============================] - 0s 34us/sample - loss: 0.5749 - acc: 0.7314
Epoch 496/500
350/350 [==============================] - 0s 40us/sample - loss: 0.5640 - acc: 0.7429
Epoch 497/500
350/350 [==============================] - 0s 37us/sample - loss: 0.5606 - acc: 0.7629
Epoch 498/500
350/350 [==============================] - 0s 43us/sample - loss: 0.5702 - acc: 0.7229
Epoch 499/500
350/350 [==============================] - 0s 46us/sample - loss: 0.5521 - acc: 0.7457
Epoch 500/500
350/350 [==============================] - 0s 40us/sample - loss: 0.5722 - acc: 0.7571
```

```
Out[126]: <tensorflow.python.keras.callbacks.History at 0x25ee0104160>
```

```
In [148]: score=model.evaluate(xtest,ytest)
```

```
150/150 [==============================] - 0s 62us/sample - loss: 0.4804 - acc: 0.8000
```

```
In [149]: s=score[1]
          s*100
```

```
Out[149]: 80.0000011920929
```

```
In [150]: import numpy as np
```

```
In [151]: a=[0,-1,2.,3]
```

```
In [152]: s=tf.nn.softmax(a)
```

In [154]:

```
with tf.Session() as sess:
    print(sess.run(s))
    #print(sess.run(a))
```

```
[0.03467109 0.01275478 0.25618663 0.69638747]
```