SIMPLE ARTIFICIAL NEURAL NETWORK MODEL FOR CLASSIFICATION

In [1]:
```
#importing relevant packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:
```
#reading the data
marks=pd.read_csv('Desktop/DATA/students_marks.csv')
```

In [3]:
```
#head of data
marks.head()
```

Out[3]:

	number_courses	time_study	Marks	Targets
0	3	4.508	19.202	0
1	4	0.096	7.734	0
2	4	3.133	13.811	0
3	6	7.909	53.018	1
4	8	7.811	55.299	1

In [4]:
```
#dtypes
marks.dtypes
```

Out[4]:
```
number_courses      int64
time_study        float64
Marks             float64
Targets             int64
dtype: object
```

In [5]:
```
#len of data
len(marks)
```

Out[5]:
```
100
```

In [6]:
```
#checking nulll values
marks.isnull().sum()
```

Out[6]:
```
number_courses    0
time_study        0
Marks             0
Targets           0
dtype: int64
```

In [7]:
```
#train test split
```

In [8]:
```
x=marks.drop('Targets',axis=1).values
y=marks['Targets'].values
```

In [9]:
```
from sklearn.model_selection import train_test_split
```

In [10]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=101)
```

In [11]:
```
y_test
```

Out[11]:
```
array([1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1,
       1, 1, 0, 0, 1, 1, 0, 1], dtype=int64)
```

In [12]:
```
#data scalling
```

In [13]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [14]:
```
scaler=MinMaxScaler()
```

In [15]:
```
scaler.fit(x_train)
```

Out[15]:
```
MinMaxScaler()
```

In [16]:
```
x_test=scaler.transform(x_test)
```

In [17]:
```
 x_train=scaler.transform(x_train)
```

In [18]:
```
# MODEL TRAINING AND TESTING
```

In [19]:
```
from tensorflow.keras.models import Sequential
```

In [20]:
```
from tensorflow.keras.layers import Dense,Dropout
```

In [21]:
```
x_train.shape
```

Out[21]:
```
(70, 3)
```

In [22]:
```
x_test.shape
```

Out[22]:
```
(30, 3)
```

In [23]:
```
y_test.shape
```

Out[23]:
```
(30,)
```

In [24]:
```
#model
```

In [25]:
```
model=Sequential()
model.add(Dense(1000,activation ='relu'))
model.add(Dense(500,activation ='relu'))
model.add(Dense(500,activation ='relu'))
model.add(Dense(1,activation ='sigmoid'))
```

In [26]:
```
model.compile(loss='binary_crossentropy',optimizer='adam',metrics='accuracy')
```

In [27]:
```
from tensorflow.keras.callbacks import EarlyStopping
```

In [28]:
```
early_stop=EarlyStopping(monitor='val_loss',mode='min',verbose=1,patience=25)
```

In [29]:
```
#model fitting
```

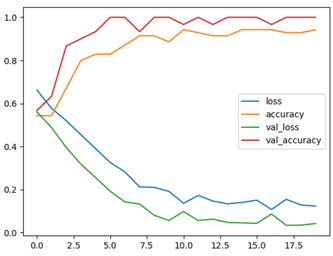
In [30]:
```
model.fit(x=x_train,y=y_train,epochs=20,batch_size=20,validation_data=(x_test,y_test),callbacks=[early_stop])
```

Out[30]:
```
Epoch 1/20
4/4 [==============================] - 1s 78ms/step - loss: 0.6622 - accuracy: 0.5429 - val_loss: 0.5604 - val_accuracy: 0.5667
Epoch 2/20
4/4 [==============================] - 0s 27ms/step - loss: 0.5773 - accuracy: 0.5429 - val_loss: 0.4872 - val_accuracy: 0.6333
Epoch 3/20
4/4 [==============================] - 0s 21ms/step - loss: 0.5194 - accuracy: 0.6714 - val_loss: 0.3952 - val_accuracy: 0.8667
Epoch 4/20
4/4 [==============================] - 0s 23ms/step - loss: 0.4544 - accuracy: 0.8000 - val_loss: 0.3176 - val_accuracy: 0.9000
Epoch 5/20
4/4 [==============================] - 0s 26ms/step - loss: 0.3899 - accuracy: 0.8286 - val_loss: 0.2553 - val_accuracy: 0.9333
Epoch 6/20
4/4 [==============================] - 0s 21ms/step - loss: 0.3255 - accuracy: 0.8286 - val_loss: 0.1915 - val_accuracy: 1.0000
Epoch 7/20
4/4 [==============================] - 0s 26ms/step - loss: 0.2817 - accuracy: 0.8714 - val_loss: 0.1424 - val_accuracy: 1.0000
Epoch 8/20
4/4 [==============================] - 0s 26ms/step - loss: 0.2121 - accuracy: 0.9143 - val_loss: 0.1324 - val_accuracy: 0.9333
Epoch 9/20
4/4 [==============================] - 0s 21ms/step - loss: 0.2096 - accuracy: 0.9143 - val_loss: 0.0799 - val_accuracy: 1.0000
Epoch 10/20
4/4 [==============================] - 0s 26ms/step - loss: 0.1911 - accuracy: 0.8857 - val_loss: 0.0559 - val_accuracy: 1.0000
Epoch 11/20
4/4 [==============================] - 0s 26ms/step - loss: 0.1357 - accuracy: 0.9429 - val_loss: 0.0972 - val_accuracy: 0.9667
Epoch 12/20
4/4 [==============================] - 0s 25ms/step - loss: 0.1722 - accuracy: 0.9286 - val_loss: 0.0558 - val_accuracy: 1.0000
Epoch 13/20
4/4 [==============================] - 0s 21ms/step - loss: 0.1460 - accuracy: 0.9143 - val_loss: 0.0622 - val_accuracy: 0.9667
Epoch 14/20
4/4 [==============================] - 0s 26ms/step - loss: 0.1333 - accuracy: 0.9143 - val_loss: 0.0465 - val_accuracy: 1.0000
Epoch 15/20
4/4 [==============================] - 0s 26ms/step - loss: 0.1399 - accuracy: 0.9429 - val_loss: 0.0448 - val_accuracy: 1.0000
Epoch 16/20
4/4 [==============================] - 0s 26ms/step - loss: 0.1501 - accuracy: 0.9429 - val_loss: 0.0426 - val_accuracy: 1.0000
Epoch 17/20
4/4 [==============================] - 0s 26ms/step - loss: 0.1069 - accuracy: 0.9429 - val_loss: 0.0858 - val_accuracy: 0.9667
Epoch 18/20
4/4 [==============================] - 0s 24ms/step - loss: 0.1542 - accuracy: 0.9286 - val_loss: 0.0335 - val_accuracy: 1.0000
Epoch 19/20
4/4 [==============================] - 0s 26ms/step - loss: 0.1278 - accuracy: 0.9286 - val_loss: 0.0339 - val_accuracy: 1.0000
Epoch 20/20
4/4 [==============================] - 0s 21ms/step - loss: 0.1224 - accuracy: 0.9429 - val_loss: 0.0416 - val_accuracy: 1.0000
<keras.src.callbacks.History at 0x2788b06d2d0>
```

In [31]:
```
losses=pd.DataFrame(model.history.history)
```

In [32]:
```
losses.plot()
```

Out[32]:
```
<Axes: >
```



In [33]:
```
pred=(model.predict(x_test)>0.5).astype('int32')
```

```
1/1 [==============================] - 0s 79ms/step
```

In [34]:
```
from sklearn.metrics import mean_squared_error
```

In [35]:
```
mean_squared_error(y_test,model.predict(x_test))
```

```
1/1 [==============================] - 0s 32ms/step
```

Out[35]:
```
0.009201021513787471
```

In [36]:
```
np.sqrt(mean_squared_error(y_test,model.predict(x_test)))
```

```
1/1 [==============================] - 0s 34ms/step
```

Out[36]:
```
0.09592195532716934
```

In [37]:
```
# model evaluation
```

In [38]:
```
from sklearn.metrics import r2_score
```

In [39]:
```
print(r2_score(y_test,pred))
```

```
1.0
```

In [40]:
```
#checking with random forest classifier
```

In [41]:
```
from sklearn.metrics import accuracy_score
```

In [42]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [43]:
```
rfc=RandomForestClassifier()
```

In [44]:
```
rfc.fit(x_train,y_train)
```

Out[44]:
```
RandomForestClassifier()
```

In [45]:
```
p=rfc.predict(x_test)
```

In [46]:
```
print(r2_score(y_test,p))
```

```
0.8642533936651583
```

CONLUSION

Artifial neural network was best with accuracy of 100% whereas Random forest classifier had 86% accuracy.