DS Lab Assignment 7

Kadali Sai Vivek

197139

Q1) Implement a Simple Neural Network Model and predict if a person will buy insurance or not for the given dataset.

In [4]:
```
from google.colab import files
uploaded=files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving insurance_data.csv to insurance_data.csv
```

In [5]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler  # For Standardizing the Data

import keras
from keras.models import Sequential   #For building the Neural Network Model layer by l
from keras.layers import Dense        # Groups linear stack of layers and used to add f

from sklearn.metrics import accuracy_score            # to determine the accuracy score
```

In [6]:
```
dataset= pd.read_csv('insurance_data.csv')
dataset
```

Out[6]:

	age	Affordability	bought_insurance
0	22	1	0
1	25	1	0
2	47	0	1
3	52	1	0
4	46	1	1
5	56	0	1
6	55	1	0
7	60	1	1
8	62	1	1
9	61	1	1
10	18	0	0
11	28	1	0
12	27	0	0

	age	Affordability	bought_insurance
13	29	0	0
14	49	1	1
15	55	1	1
16	25	0	1
17	58	0	1
18	19	1	0
19	18	1	0
20	21	1	0
21	26	0	0
22	40	1	1
23	45	0	1
24	50	1	1
25	54	1	1
26	23	1	0

In [7]:

```
# Number of Columns
n=dataset.shape[1]
# Number of Rows
m=dataset.shape[0]

# Dividing Dataset into Training and Testing (80% -20%)
x_train=np.array(dataset.iloc[:int(0.8*m),:n-1])
y_train=np.array(dataset.iloc[:int(0.8*m),-1])
x_test=np.array(dataset.iloc[int(0.8*m):,:n-1])
y_test=np.array(dataset.iloc[int(0.8*m):,-1])


# Standardizing the training and testing data using standardscaler()
SC = StandardScaler()

x_train_scaled= SC.fit_transform(x_train)
x_test_scaled=SC.fit_transform(x_test)

model=Sequential()                                          # Create Neural Network Model
model.add(Dense(2, input_dim=2, activation='relu'))     # Input Dimension in this layer
model.add(Dense(2, input_dim=2, activation='relu'))     # Input Dimension in this layer
model.add(Dense(1, activation='softmax'))               # Output dimension has only 1 p


# Adam is the optimizer that can perform the gradient descent and accelerate the gradie
# loss function as binary_crossentropy as we have 2 cases only 0 or 1
model.compile(optimizer='adam', loss='binary_crossentropy',metrics=['accuracy'])


model.fit(x_train_scaled, y_train, epochs=100,batch_size=25) # Fit the model
model.evaluate(x_test_scaled,y_test)                        # Evalvuate using scaled te
```

```
y_pred=model.predict(x_test_scaled)                          # Predict based on scaled t


#Calculating Accuracy of the Model
actual=y_test.tolist()
prediction=y_pred.tolist()
a = accuracy_score(prediction,actual)                      # Used accuracy_score functio
print('Accuracy of the Model is :', a*100)
```

```
Epoch 1/100
1/1 [==============================] - 1s 657ms/step - loss: 0.5955 - accuracy: 0.4762
Epoch 2/100
1/1 [==============================] - 0s 12ms/step - loss: 0.5949 - accuracy: 0.4762
Epoch 3/100
1/1 [==============================] - 0s 9ms/step - loss: 0.5943 - accuracy: 0.4762
Epoch 4/100
1/1 [==============================] - 0s 12ms/step - loss: 0.5937 - accuracy: 0.4762
Epoch 5/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5931 - accuracy: 0.4762
Epoch 6/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5925 - accuracy: 0.4762
Epoch 7/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5919 - accuracy: 0.4762
Epoch 8/100
1/1 [==============================] - 0s 7ms/step - loss: 0.5913 - accuracy: 0.4762
Epoch 9/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5907 - accuracy: 0.4762
Epoch 10/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5901 - accuracy: 0.4762
Epoch 11/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5894 - accuracy: 0.4762
Epoch 12/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5887 - accuracy: 0.4762
Epoch 13/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5880 - accuracy: 0.4762
Epoch 14/100
1/1 [==============================] - 0s 7ms/step - loss: 0.5873 - accuracy: 0.4762
Epoch 15/100
1/1 [==============================] - 0s 7ms/step - loss: 0.5866 - accuracy: 0.4762
Epoch 16/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5859 - accuracy: 0.4762
Epoch 17/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5852 - accuracy: 0.4762
Epoch 18/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5845 - accuracy: 0.4762
Epoch 19/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5839 - accuracy: 0.4762
Epoch 20/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5832 - accuracy: 0.4762
Epoch 21/100
1/1 [==============================] - 0s 19ms/step - loss: 0.5825 - accuracy: 0.4762
Epoch 22/100
1/1 [==============================] - 0s 9ms/step - loss: 0.5818 - accuracy: 0.4762
Epoch 23/100
1/1 [==============================] - 0s 7ms/step - loss: 0.5811 - accuracy: 0.4762
Epoch 24/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5805 - accuracy: 0.4762
Epoch 25/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5798 - accuracy: 0.4762
Epoch 26/100
```

```
1/1 [==============================] - 0s 11ms/step - loss: 0.5792 - accuracy: 0.4762
Epoch 27/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5785 - accuracy: 0.4762
Epoch 28/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5779 - accuracy: 0.4762
Epoch 29/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5772 - accuracy: 0.4762
Epoch 30/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5766 - accuracy: 0.4762
Epoch 31/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5760 - accuracy: 0.4762
Epoch 32/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5753 - accuracy: 0.4762
Epoch 33/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5747 - accuracy: 0.4762
Epoch 34/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5741 - accuracy: 0.4762
Epoch 35/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5735 - accuracy: 0.4762
Epoch 36/100
1/1 [==============================] - 0s 7ms/step - loss: 0.5729 - accuracy: 0.4762
Epoch 37/100
1/1 [==============================] - 0s 4ms/step - loss: 0.5723 - accuracy: 0.4762
Epoch 38/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5717 - accuracy: 0.4762
Epoch 39/100
1/1 [==============================] - 0s 8ms/step - loss: 0.5711 - accuracy: 0.4762
Epoch 40/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5705 - accuracy: 0.4762
Epoch 41/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5700 - accuracy: 0.4762
Epoch 42/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5694 - accuracy: 0.4762
Epoch 43/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5688 - accuracy: 0.4762
Epoch 44/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5683 - accuracy: 0.4762
Epoch 45/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5677 - accuracy: 0.4762
Epoch 46/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5672 - accuracy: 0.4762
Epoch 47/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5666 - accuracy: 0.4762
Epoch 48/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5661 - accuracy: 0.4762
Epoch 49/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5655 - accuracy: 0.4762
Epoch 50/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5650 - accuracy: 0.4762
Epoch 51/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5645 - accuracy: 0.4762
Epoch 52/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5640 - accuracy: 0.4762
Epoch 53/100
1/1 [==============================] - 0s 9ms/step - loss: 0.5634 - accuracy: 0.4762
Epoch 54/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5629 - accuracy: 0.4762
Epoch 55/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5624 - accuracy: 0.4762
Epoch 56/100
```

```
1/1 [==============================] - 0s 5ms/step - loss: 0.5619 - accuracy: 0.4762
Epoch 57/100
1/1 [==============================] - 0s 8ms/step - loss: 0.5614 - accuracy: 0.4762
Epoch 58/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5609 - accuracy: 0.4762
Epoch 59/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5605 - accuracy: 0.4762
Epoch 60/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5600 - accuracy: 0.4762
Epoch 61/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5595 - accuracy: 0.4762
Epoch 62/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5590 - accuracy: 0.4762
Epoch 63/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5585 - accuracy: 0.4762
Epoch 64/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5581 - accuracy: 0.4762
Epoch 65/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5576 - accuracy: 0.4762
Epoch 66/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5572 - accuracy: 0.4762
Epoch 67/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5567 - accuracy: 0.4762
Epoch 68/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5563 - accuracy: 0.4762
Epoch 69/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5558 - accuracy: 0.4762
Epoch 70/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5554 - accuracy: 0.4762
Epoch 71/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5549 - accuracy: 0.4762
Epoch 72/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5545 - accuracy: 0.4762
Epoch 73/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5541 - accuracy: 0.4762
Epoch 74/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5536 - accuracy: 0.4762
Epoch 75/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5532 - accuracy: 0.4762
Epoch 76/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5528 - accuracy: 0.4762
Epoch 77/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5524 - accuracy: 0.4762
Epoch 78/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5520 - accuracy: 0.4762
Epoch 79/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5516 - accuracy: 0.4762
Epoch 80/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5512 - accuracy: 0.4762
Epoch 81/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5508 - accuracy: 0.4762
Epoch 82/100
1/1 [==============================] - 0s 4ms/step - loss: 0.5504 - accuracy: 0.4762
Epoch 83/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5500 - accuracy: 0.4762
Epoch 84/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5496 - accuracy: 0.4762
Epoch 85/100
1/1 [==============================] - 0s 5ms/step - loss: 0.5492 - accuracy: 0.4762
Epoch 86/100
```

```
1/1 [==============================] - 0s 17ms/step - loss: 0.5488 - accuracy: 0.4762
Epoch 87/100
1/1 [==============================] - 0s 8ms/step - loss: 0.5484 - accuracy: 0.4762
Epoch 88/100
1/1 [==============================] - 0s 7ms/step - loss: 0.5480 - accuracy: 0.4762
Epoch 89/100
1/1 [==============================] - 0s 8ms/step - loss: 0.5476 - accuracy: 0.4762
Epoch 90/100
1/1 [==============================] - 0s 7ms/step - loss: 0.5473 - accuracy: 0.4762
Epoch 91/100
1/1 [==============================] - 0s 8ms/step - loss: 0.5469 - accuracy: 0.4762
Epoch 92/100
1/1 [==============================] - 0s 9ms/step - loss: 0.5465 - accuracy: 0.4762
Epoch 93/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5462 - accuracy: 0.4762
Epoch 94/100
1/1 [==============================] - 0s 8ms/step - loss: 0.5458 - accuracy: 0.4762
Epoch 95/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5454 - accuracy: 0.4762
Epoch 96/100
1/1 [==============================] - 0s 7ms/step - loss: 0.5451 - accuracy: 0.4762
Epoch 97/100
1/1 [==============================] - 0s 7ms/step - loss: 0.5447 - accuracy: 0.4762
Epoch 98/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5444 - accuracy: 0.4762
Epoch 99/100
1/1 [==============================] - 0s 6ms/step - loss: 0.5440 - accuracy: 0.4762
Epoch 100/100
1/1 [==============================] - 0s 4ms/step - loss: 0.5437 - accuracy: 0.4762
1/1 [==============================] - 0s 276ms/step - loss: 0.4390 - accuracy: 0.6667
Accuracy of the Model is : 66.66666666666666
```

Q2) Use the given dataset and build a customer churn prediction model using artificial neural network.

In [8]:
```
from google.colab import files
uploaded=files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Churn_Modelling.csv to Churn_Modelling.csv
```

In [9]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler  # For Standardizing the Data
```

In [10]:
```
# importing the dataset
dataset= pd.read_csv('Churn_Modelling.csv')
dataset
```

Out[10]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance
0	1	15634602	Hargrave	619	France	Female	42	2	0.00
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance
2	3	15619304	Onio	502	France	Female	42	8	159660.80
3	4	15701354	Boni	699	France	Female	39	1	0.00
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82
...	...	...	...	...	...	...	...	...	...
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61
9997	9998	15584532	Liu	709	France	Female	36	7	0.00
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79

10000 rows × 14 columns

In [11]:

```
# Drop 1st 3 Columns as we don't need them
print("Drop RowNumber ,CustomerID and Surname from Data")
dataset.drop(dataset.columns[[0,1,2]],axis=1,inplace=True)
dataset
```

```
Drop RowNumber ,CustomerID and Surname from Data
```

Out[11]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveM
0	619	France	Female	42	2	0.00	1	1	
1	608	Spain	Female	41	1	83807.86	1	0	
2	502	France	Female	42	8	159660.80	3	1	
3	699	France	Female	39	1	0.00	2	0	
4	850	Spain	Female	43	2	125510.82	1	1	
...	...	...	...	...	...	...	...	...	...
9995	771	France	Male	39	5	0.00	2	1	
9996	516	France	Male	35	10	57369.61	1	1	
9997	709	France	Female	36	7	0.00	1	0	
9998	772	Germany	Male	42	3	75075.31	2	1	
9999	792	France	Female	28	4	130142.79	1	1	

10000 rows × 11 columns

In [12]:

```
# Number of Columns
n=dataset.shape[1]
# Number of Rows
```

```
m=dataset.shape[0]


def Convert_text_to_int(Array):
  visited=[]
  res=[]
  index=0
  for i in Array:
    if(i in visited):
      res.append(visited.index(i))
    else:
      visited.append(i)
      res.append(index)
      index=index+1
  return res

# converting the text data in geography and gender column into Integer Data
dataset.iloc[:,1]=np.array(Convert_text_to_int(dataset.iloc[:,1].tolist()))
dataset.iloc[:,2]=np.array(Convert_text_to_int(dataset.iloc[:,2].tolist()))
dataset
```

Out[12]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveM
0	619	0	0	42	2	0.00	1	1	
1	608	1	0	41	1	83807.86	1	0	
2	502	0	0	42	8	159660.80	3	1	
3	699	0	0	39	1	0.00	2	0	
4	850	1	0	43	2	125510.82	1	1	
...	...	...	...	...	...	...	...	...	
9995	771	0	1	39	5	0.00	2	1	
9996	516	0	1	35	10	57369.61	1	1	
9997	709	0	0	36	7	0.00	1	0	
9998	772	2	1	42	3	75075.31	2	1	
9999	792	0	0	28	4	130142.79	1	1	

10000 rows × 11 columns

In [15]:

```
# Dividing Dataset into Training and Testing (80% -20%)
x_train=np.array(dataset.iloc[:int(0.8*m),:n-1])
y_train=np.array(dataset.iloc[:int(0.8*m),-1])
x_test=np.array(dataset.iloc[int(0.8*m):,:n-1])
y_test=np.array(dataset.iloc[int(0.8*m):,-1])


# Standardizing the training and testing data using standardscaler()
SC = StandardScaler()

x_train_scaled= SC.fit_transform(x_train)
x_test_scaled=SC.fit_transform(x_test)
```

```
model=Sequential()                                          # Create Neural Network Model
model.add(Dense(10, input_dim=10, activation='relu'))       # Input dimensions to this laye
model.add(Dense(6, input_dim=10, activation='relu'))        # input dimensions to this laye
model.add(Dense(4, input_dim=6, activation='relu'))         # input dimensions to this laye
model.add(Dense(1, activation='sigmoid'))                   # Output dimension has only 1 p


# Adam is the optimizer that can perform the gradient descent and accelerate the gradie
# loss function as binary_crossentropy as we have 2 cases only 0 or 1
model.compile(optimizer='adam', loss='binary_crossentropy',metrics=['accuracy'])


# Fit the Model
model.fit(x_train_scaled, y_train, epochs=100,batch_size=250)
model.evaluate(x_test_scaled,y_test)                                # Evaluate using our scale
y_pred=model.predict(x_test_scaled)                                 # Predict based on scaled


#Calculating Accuracy of the Model
actual=y_test.tolist()
prediction=y_pred.tolist()
for i in range(len(prediction)):                            # since we used sigmoid function
  if(prediction[i][0]>=0.5):                                # we convert those values into eith
    prediction[i]=1                                         # if greater or equal to 0.5 we mak
  else:
    prediction[i]=0                                         # if less than 0.5 we make it equal

a = accuracy_score(prediction,actual)
print('Accuracy of Model is :', a*100)
```

```
Epoch 1/100
32/32 [==============================] - 1s 4ms/step - loss: 0.7716 - accuracy: 0.2269
Epoch 2/100
32/32 [==============================] - 0s 3ms/step - loss: 0.7100 - accuracy: 0.4619
Epoch 3/100
32/32 [==============================] - 0s 4ms/step - loss: 0.6846 - accuracy: 0.6896
Epoch 4/100
32/32 [==============================] - 0s 4ms/step - loss: 0.6701 - accuracy: 0.7592
Epoch 5/100
32/32 [==============================] - 0s 4ms/step - loss: 0.6594 - accuracy: 0.7847
Epoch 6/100
32/32 [==============================] - 0s 5ms/step - loss: 0.6504 - accuracy: 0.7916
Epoch 7/100
32/32 [==============================] - 0s 3ms/step - loss: 0.6423 - accuracy: 0.7941
Epoch 8/100
32/32 [==============================] - 0s 6ms/step - loss: 0.6347 - accuracy: 0.7946
Epoch 9/100
32/32 [==============================] - 0s 4ms/step - loss: 0.6273 - accuracy: 0.7949
Epoch 10/100
32/32 [==============================] - 0s 4ms/step - loss: 0.6200 - accuracy: 0.7958
Epoch 11/100
32/32 [==============================] - 0s 5ms/step - loss: 0.6122 - accuracy: 0.7984
Epoch 12/100
32/32 [==============================] - 0s 5ms/step - loss: 0.6031 - accuracy: 0.8023
Epoch 13/100
32/32 [==============================] - 0s 3ms/step - loss: 0.5887 - accuracy: 0.8076
Epoch 14/100
```

```
32/32 [==============================] - 0s 4ms/step - loss: 0.5594 - accuracy: 0.8114
Epoch 15/100
32/32 [==============================] - 0s 8ms/step - loss: 0.5107 - accuracy: 0.8154
Epoch 16/100
32/32 [==============================] - 0s 7ms/step - loss: 0.4591 - accuracy: 0.8196
Epoch 17/100
32/32 [==============================] - 0s 5ms/step - loss: 0.4292 - accuracy: 0.8209
Epoch 18/100
32/32 [==============================] - 0s 6ms/step - loss: 0.4189 - accuracy: 0.8244
Epoch 19/100
32/32 [==============================] - 0s 8ms/step - loss: 0.4140 - accuracy: 0.8267
Epoch 20/100
32/32 [==============================] - 0s 6ms/step - loss: 0.4105 - accuracy: 0.8282
Epoch 21/100
32/32 [==============================] - 0s 6ms/step - loss: 0.4072 - accuracy: 0.8313
Epoch 22/100
32/32 [==============================] - 0s 5ms/step - loss: 0.4040 - accuracy: 0.8325
Epoch 23/100
32/32 [==============================] - 0s 6ms/step - loss: 0.4004 - accuracy: 0.8363
Epoch 24/100
32/32 [==============================] - 0s 6ms/step - loss: 0.3971 - accuracy: 0.8380
Epoch 25/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3937 - accuracy: 0.8405
Epoch 26/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3897 - accuracy: 0.8415
Epoch 27/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3860 - accuracy: 0.8430
Epoch 28/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3818 - accuracy: 0.8435
Epoch 29/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3778 - accuracy: 0.8441
Epoch 30/100
32/32 [==============================] - 0s 2ms/step - loss: 0.3735 - accuracy: 0.8471
Epoch 31/100
32/32 [==============================] - 0s 6ms/step - loss: 0.3695 - accuracy: 0.8503
Epoch 32/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3657 - accuracy: 0.8509
Epoch 33/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3620 - accuracy: 0.8509
Epoch 34/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3587 - accuracy: 0.8528
Epoch 35/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3558 - accuracy: 0.8553
Epoch 36/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3538 - accuracy: 0.8549
Epoch 37/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3515 - accuracy: 0.8558
Epoch 38/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3497 - accuracy: 0.8571
Epoch 39/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3484 - accuracy: 0.8574
Epoch 40/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3468 - accuracy: 0.8565
Epoch 41/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3459 - accuracy: 0.8581
Epoch 42/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3447 - accuracy: 0.8575
Epoch 43/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3440 - accuracy: 0.8574
Epoch 44/100
```

```
32/32 [==============================] - 0s 7ms/step - loss: 0.3433 - accuracy: 0.8570
Epoch 45/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3429 - accuracy: 0.8565
Epoch 46/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3429 - accuracy: 0.8568
Epoch 47/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3415 - accuracy: 0.8574
Epoch 48/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3415 - accuracy: 0.8580
Epoch 49/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3405 - accuracy: 0.8580
Epoch 50/100
32/32 [==============================] - 0s 6ms/step - loss: 0.3404 - accuracy: 0.8577
Epoch 51/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3400 - accuracy: 0.8590
Epoch 52/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3398 - accuracy: 0.8569
Epoch 53/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3389 - accuracy: 0.8583
Epoch 54/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3387 - accuracy: 0.8575
Epoch 55/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3383 - accuracy: 0.8577
Epoch 56/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3385 - accuracy: 0.8575
Epoch 57/100
32/32 [==============================] - 0s 2ms/step - loss: 0.3381 - accuracy: 0.8602
Epoch 58/100
32/32 [==============================] - 0s 7ms/step - loss: 0.3376 - accuracy: 0.8589
Epoch 59/100
32/32 [==============================] - 0s 6ms/step - loss: 0.3372 - accuracy: 0.8599
Epoch 60/100
32/32 [==============================] - 0s 6ms/step - loss: 0.3369 - accuracy: 0.8594
Epoch 61/100
32/32 [==============================] - 0s 7ms/step - loss: 0.3371 - accuracy: 0.8600
Epoch 62/100
32/32 [==============================] - 0s 8ms/step - loss: 0.3366 - accuracy: 0.8589
Epoch 63/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3362 - accuracy: 0.8590
Epoch 64/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3361 - accuracy: 0.8600
Epoch 65/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3364 - accuracy: 0.8609
Epoch 66/100
32/32 [==============================] - 0s 10ms/step - loss: 0.3358 - accuracy: 0.8590
Epoch 67/100
32/32 [==============================] - 0s 8ms/step - loss: 0.3353 - accuracy: 0.8599
Epoch 68/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3353 - accuracy: 0.8605
Epoch 69/100
32/32 [==============================] - 0s 7ms/step - loss: 0.3346 - accuracy: 0.8606
Epoch 70/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3345 - accuracy: 0.8609
Epoch 71/100
32/32 [==============================] - 0s 6ms/step - loss: 0.3346 - accuracy: 0.8608
Epoch 72/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3341 - accuracy: 0.8608
Epoch 73/100
32/32 [==============================] - 0s 6ms/step - loss: 0.3337 - accuracy: 0.8611
Epoch 74/100
```

```
32/32 [==============================] - 0s 4ms/step - loss: 0.3337 - accuracy: 0.8614
Epoch 75/100
32/32 [==============================] - 0s 6ms/step - loss: 0.3338 - accuracy: 0.8599
Epoch 76/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3339 - accuracy: 0.8619
Epoch 77/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3333 - accuracy: 0.8606
Epoch 78/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3330 - accuracy: 0.8611
Epoch 79/100
32/32 [==============================] - 0s 6ms/step - loss: 0.3329 - accuracy: 0.8610
Epoch 80/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3328 - accuracy: 0.8610
Epoch 81/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3324 - accuracy: 0.8618
Epoch 82/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3324 - accuracy: 0.8616
Epoch 83/100
32/32 [==============================] - 0s 6ms/step - loss: 0.3324 - accuracy: 0.8616
Epoch 84/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3318 - accuracy: 0.8625
Epoch 85/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3324 - accuracy: 0.8622
Epoch 86/100
32/32 [==============================] - 0s 6ms/step - loss: 0.3317 - accuracy: 0.8622
Epoch 87/100
32/32 [==============================] - 0s 6ms/step - loss: 0.3319 - accuracy: 0.8614
Epoch 88/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3313 - accuracy: 0.8611
Epoch 89/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3313 - accuracy: 0.8621
Epoch 90/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3317 - accuracy: 0.8629
Epoch 91/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3316 - accuracy: 0.8627
Epoch 92/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3311 - accuracy: 0.8631
Epoch 93/100
32/32 [==============================] - 0s 2ms/step - loss: 0.3314 - accuracy: 0.8634
Epoch 94/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3307 - accuracy: 0.8633
Epoch 95/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3308 - accuracy: 0.8631
Epoch 96/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3303 - accuracy: 0.8622
Epoch 97/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3301 - accuracy: 0.8634
Epoch 98/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3303 - accuracy: 0.8629
Epoch 99/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3306 - accuracy: 0.8636
Epoch 100/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3301 - accuracy: 0.8637
63/63 [==============================] - 1s 4ms/step - loss: 0.3589 - accuracy: 0.8520
Accuracy of Model is : 85.2
```